Guideline for Conformance to Common Data Format

Version 1.0

February 28, 2023

Japan Analytical Instruments Manufacturers' Association

Contents

1. Introduction

This guideline was prepared by the Japan Analytical Instruments Manufacturers' Association (JAIMA) as a guideline that can be used when indicating analytical/measuring results or protocols using the MaiML[1] common data format for analytical and measuring instruments specified in **JIS K 0200** (hereinafter "**JIS**").

MaiML is a data format that conforms to XML (Extensible Markup Language[2]) format specifications. MaiML specifications were prepared to enable modeling measurement, analysis, and related processes and collectively expressing them together with measurement data.

This guideline was prepared assuming the MaiML data format would be used in the following types of situations. In other words, content can be indicated using the MaiML data format and corresponding MaiML files can be used for data analysis in the following situations. This guideline can also be used to determine the intention and meaning of tag elements specified in the MaiML data format and to create the means and methods for utilizing tag elements.

- Corporate personnel, researchers, or others involved in the development (or manufacturing) of measuring or analytical instruments or software using MaiML to indicate measuring/analytical results or protocols (hereinafter "case A").
- Development personnel, researchers, or others involved in the development (or manufacturing) of new materials or other items using MaiML to indicate results or protocols from measuring/analyzing such materials/items (hereinafter "case B").
- Using MaiML to indicate measurement/analysis sample preparation protocols or preparation process results related to the development of measuring or analytical instruments or the development of new materials (hereinafter "case C").

This guideline provides supplemental explanatory information for examples indicated in the **JIS** standard where the purpose for using MaiML may be difficult to understand, describes additional use cases for MaiML, lists frequently asked questions about using MaiML, and so on. It also includes an index of terminology used in the discussion of those examples, MaiML use cases, questions, and so on, and also cites corresponding standards. The purpose is to promote a deeper understanding of the **JIS** standard.

In this Guideline, chapter **2** provides an overview of the approach and content of MaiML, chapter **3** describes an overview of the XML code and Petri nets used for MaiML expressions, chapter 4 describes the content of MaiML files, chapters **5** to **7** provide examples of indicating data, citing external files, and specific methods for indicating confidentiality, safety, and other characteristics, chapter **8** provides an overview of terminology, and chapters **9** and **10** describe specific examples. Furthermore, common questions and concerns are listed in chapter **11**.

Note that a MaiML schema has been created for verifying that data formats created based on this guideline conforms to **JIS** standards.

1 MaiML: Abbreviation for "Measurement Analysis Instrument Markup Language." It is a language for writing data in process steps, measurement/analysis results, logs, or other data associated with analytical and measuring devices or software.
[2] XML is a markup language used to indicate the structure and content of data in files and serves as the industry standard for sending content (data) via the Internet.

2. Overview

2.1 Intended Approach for the MaiML Data Format

MaiML is a language for expressing all input, process, and output data related to measurements and analysis (hereinafter “measurement/analysis data”). It conforms to XML (Extensible Markup Language) standards. Though referred to as a language, it is not a programming language intended for processes, but rather a language for indicating the content of data. The data format based on the MaiML language is referred to as the MaiML data format. Files[3] written according to the MaiML data format are referred to as MaiML files. The purpose of MaiML is to enable the expression of all data related to measurements and analysis.

If a MaiML file is used after its measurement/analysis data has been registered in cyberspace, then only the information registered in cyberspace is included. That ensures that MaiML files contain all the relevant information necessary for the corresponding measurement/analysis data. The ability to express everything relevant to a particular measurement/analysis in the measurement/analysis data space is referred to as independent availability[4].

In response to recent changes in the use of big data, the following factors were particularly considered when specifying MaiML data format requirements.

1 Data Readability
  1.1 The overall structure is based on file units of text rather than binary code. All files with XML code are written using text. That is one reason XML was selected as a method for indicating data.
  1.2 On the other hand, to ensure data can be handled quickly, the ability to use external files archived using the ZIP format was included to provide compatibility with binary files and generic formats.
  1.3 With files as units of data, MaiML can also be used to distribute data via networks.
2 Using Specified File Formats
  2.1 To ensure commonly available generic tools can be used, files must be used in accordance with the specified formats and conventions. That is one reason why MaiML was made in conformance to and based on XML, PNML, and XES languages.
3 Structuring Data (sections **2.1.2**, **2.1.3**, and **3.1**)
  3.1 MaiML enables all data output from various measuring/analytical devices, software, or processes related to measurement/analysis to be packaged as file units in cyberspace or a data lake[5].
  3.2 The content of files in a data lake is indicated using an XML-compatible markup language and is structured so that it can be extracted as a single set of measurement/analysis terminology, data type, and data values.
  3.3 Data can be accessed using generic development tools.
4 Modeling Measuring/Analytical Processes and Configuring Petri Nets (sections **2.1.6.1** and **3.3**)
  4.1 Measurement/analysis processes can be modeled as discrete events, such as measuring/analytical operations, or related operations or the processes can be

[3]Files: In information systems, files are generally used to indicate a unit of data, which is also the case for data indicated using MaiML. In other words, MaiML files are the smallest unit of measurement/analysis data recorded in a hard drive or other external storage devices and the unit of measurement/analysis data sent or received in communications between computers.
[4] Independent availability level: The level of independent availability that MaiML files should have will continue to be discussed.
[5] Data lake: In general, a data lake refers to a central data repository that enables data collected from a variety of sources to be saved in one location. In this context, it refers to a collection of data output from a variety of analytical/measuring devices or software. However, to ensure the various types of analytical/measuring data are stored in a form that can be used whenever needed, the MaiML data format was established to specify minimum data formatting and processing requirements for ensuring compatibility with measurements and analysis.

represented as a Petri net configured using some of the functions specified by the Petri net markup language (PNML).
That is because the flow of events is important for expressing combined measuring/analytical processes, such as those that include additional material manufacturing, pretreatment, or post-measurement analysis steps, or additional analysis based on analytical results.

5 Measurement/Analysis Terminology and Namespaces (section **2.1.4**)
- 5.1 To ensure MaiML can be used after the combination, evolution, or advancement of measurements/analysis, requirements are not specified for measurement/analysis terminology itself.
- 5.2 On the other hand, to ensure that the individuals or groups that create MaiML files use terminology clearly, the individuals or groups that use the measurement/analysis terminology or other elements shall use the designated namespace to indicate the terminology used.

6 Independent Availability
- 6.1 To ensure independent availability, MaiML permits linking to external files or files specified by a URI value.
- 6.2 To enable commercial and other data transactions, while also expecting to include as much data as possible, MaiML permits applying confidentiality to only certain portions of files.

7 Data Uniqueness (section **3.2**) and Tampering Prevention (Chapter **7**)
- 7.1 UUID (universally unique identifier) values are used to ensure the uniqueness of data in cyberspace and data lakes. That ensures that any data can be uniquely identified even after it is combined with any other data.
- 7.2 Electronic signatures and links between multiple files shall be used to prevent data tampering.

8 Traceability[6] and the XES (Extensible Event Stream) Standard Format[7]
- 8.1 To enable traceability using information such as the data creation time, data creator name, and the event log concept indicated by XES, information related to measurement/analysis operations (events) is indicated using some functionality for the XES Standard format, MaiML general purpose data containers, and other elements.

2.1.1 How to Use MaiML Files

MaiML files can also be used as groups (sets) of data to be processed by a converter or communications protocol for sending data back and forth between measurement/analysis instruments or related software. For example, presumably it will be possible to use MaiML files as a common protocol for communicating measuring/analytical instrument data between databases.

MaiML file use cases are described in section **2.2** and chapter **10**.

Furthermore, if MaiML files are registered in cyberspace, then UUID (Universally Unique Identifier) values are used to ensure the uniqueness of multiple MaiML files. In addition to uniquely indicating <maiml> elements,

[6]Traceability: Traceability is a term specified in **JIS Q 9000** Quality management systems — Fundamentals and vocabulary that refers to the ability to trace/track measurements/analyses and data analysis process steps, including revision histories. That means traceable information, such as the date/time and methods for perform measurements/analyses and data analysis, must be included in MaiML files. Meanwhile, the term "traceability" can also be used based on a different meaning for weighing and measuring applications, as defined in **ISO/IEC Guide 99**. That can cause some confusion. Therefore, to avoid the confusion, the Japanese version of this regulation refers to traceability using the term "Tsuiseki Kanosei" (追跡可能性), which is the Japanese translation of "traceability."
[7] XES Standard format: An XML-compliant format specified by the IEEE standards association that enables information system designers to determine system behavior based on event log or other information.

UUIDs are also used to specify structural content indicated within <maiml> elements. That means the uniqueness of respective elements can be indicated even if a combination of files is used in cyberspace or elsewhere. The meaning of uniqueness is defined in detail with respect to global elements in section **3.2**.

2.1.2 MaiML Files as XML Files

The purpose of MaiML is to enable the expression of all input, process, and output data related to measurements and analysis. The reasons include the following.

1. Readability: The language is defined using the text format.
2. Structuring: Due to the hierarchical indication of tag names, attributes, and corresponding content, data can be easily structured.
3. Expandability: MaiML can be easily extended by specifying tag names.
4. Guaranteeing validity: The XML schema can be used to validate expanded formats.
5. Confidentiality: Portions of files can be kept confidential.
6. Tampering prevention: MaiML includes a system for applying electronic signatures.
7. Compatibility with other formats: Data based on other XML-compliant formats can be easily input.
8. Availability of generic tools: Since it has been about 20 years since the XML standard was established, there are many development and analysis tools available.

The tags used to indicate content in the XML format can be used to express the meaning and hierarchical structure of data. Therefore, the data type information necessary for efficiently exchanging data between different programs or systems can be indicated.

Conformance to XML 1.0 has been shown, which processes files starting with the following expression.

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

Note: “Encoding” refers to the character codes used in MaiML files. Normally, UTF-8 coding is used.

MaiML indicates data as shown below, using <maiml> elements to indicate unified sets of data (files) for exchanging measurement/analysis data from the entire series of process steps and indicated with input, process, and output information.

```
<maiml version="1.0" xmlns="http://www.maiml.org/schemas"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:type="maimlRootType">

   <!-- Includes all content specified by the MaiML data format -->

</maiml>
```

For storage devices, MaiML processes files as units of data. In such cases, the file extension for MaiML files is either “.maiml” or “.mai.”

Files located on multiple external devices (either local or in cyberspace) (hereinafter “external files”) can be indicated in terms of URL values. Bundled external files are treated as archive files combined by the ZIP format. In that case, the file extension “.maiml.zip” is used. External files designated as local are expected to be bundled.

However, MaiML permits external files designated as local and cited by <insertion>, <chain>, or <parent> elements to not be archived in ZIP format. That is because it is assumed such files will not or cannot be disclosed. Such elements can be used to determine whether or not an external file exists in any of those elements.

In that case, if an external file is indicated by a MaiML file, then uniqueness is ensured by the <uuid> and <hash> element values. In the case of files with other formats, the <hash> element value contains a file digest value that mostly ensures uniqueness. Currently, files with different content and digest values with at least as many bits as SHA-2, could in principle have identical digest values, but in actuality no such cases have been indicated. That makes it possible to guarantee that the file is unique and, in other words, not tampered with or modified.

2.1.3 Structure of MaiML Files

The MaiML data format specifies 0 to 4 hierarchy levels, as indicated in **JIS K 0200**, section **5.1**, and defines the overall file composition (Clause **5**). The element names for tags used to indicate the basic structure of MaiML files are defined to indicate the intended structure of that data. That is described in more detail separately.

Level 0 tags, with the element name <maiml>, are used to span the entire MaiML file and indicate the file range.

Level 1 is defined by four types of tags with the element names <document>, <protocol>, <data>, and <eventLog>.

Though tags for level 2 and lower levels are defined separately, MaiML defines these tags for structuring measurement/analysis processes.

Specific data, metadata values/types, and other metadata are indicated using general purpose data containers. Data and various values required by respective measurement/analysis instruments are defined using <property> and <content> tags that indicate general purpose data containers. That is described in more detail separately.

MaiML can express either local files in ZIP format, files present in local directories, or files located on a network (external files indicated above), which are specified using URI values. That is defined in more detail separately.

2.1.4 Unification of Measurement/Analysis Terminology Used in Namespaces and MaiML Files

MaiML files are used to model measurement/analysis processes and only specify the general purpose data containers (section **2.1.4**) used to indicate the elements and data that determine the structures used to indicate those processes. The <name> element is used to indicate terminology in elements that indicate structure, whereas key attributes are used to indicate terminology in general purpose data containers.

Normally, it may seem more natural for respective vendors or developers to specify measurement/analysis terminology used within files created by measurement/analysis instruments or software, as seen in the case of the AniML and other data formats.

In contrast, MaiML does not specify measurement/analysis terminology used within the MaiML files created by measurement/analysis instruments or software.

That is because of the progress already made toward unifying and defining the measurement/analysis instrument and software terminology used for each method of measurement/analysis. In the case of MaiML, the terminology for XML identifiers used for the <name> element and key attributes still needs to be determined. By permitting terminology similar to natural language, the intention was to indicate a procedure for defining terminology used as XML identifiers. Furthermore, it was also intended to integrate new terminology in response to the ongoing integration of measurement/analysis instruments and development of new techniques.

Considering that in addition to measurement/analysis terminology defined for respective measurement/analysis methods, terminology and variants defined independently by vendors are also used to express measurement/analysis data, terminology differences between vendors can cause problems. However, such confusion caused by additions or variations to terminology can be avoided by differentiating between terminology using the vendor-specific namespace specified in XML.

Furthermore, the definitions of terminology and the relationships between terms can be indicated based on how ontology is indicated in an RDF (Resource Description Framework) using representative XML, which can be used to express the equivalence, similarity, or other characteristics of terms. That technique is described in the chapter on how to indicate ontology (Chapter **8**).

2.1.5 General Purpose Data Containers for Indicating Measurement/Analysis Data

In MaiML files, <property>, <content>, and <uncertainty> elements are used as general purpose data containers for expressing measurement/analysis-related data.

Data containers are grouped based on elements that indicate MaiML structure, with the meaning (name) of each data value defined by key attributes. The purpose is to enable vendors to freely name and extract any data included in the original measurement/analysis data, regardless of any interest in the meaning of each data value.

Each data type is defined by using the xsi:type attribute to specify **Tables 24** to **26** in **JIS K 0200** section **7.1.3**. These data types describe data using <value> elements, which are child elements and general purpose data containers. The precision or other characteristics of values are expressed based on the formatString attribute and the value unit by the units attribute.

If, for example, parameter compatibility is significantly compromised, such as by supporting a new measuring technique or data analysis method in a new version of an instrument or software, even by the same vendor, then the MaiML format can individually differentiate between measurement/analysis data from the new/old version of instruments or software by assigning new namespace URLs for the new and old versions of the instrument or software.

2.1.6 Relevance to Other Data Formats

2.1.6.1 PNML Format

Below <protocol> elements, some of the elements and attributes defined in PNML (Petri net markup language) are used to express the process flow of measurement/analysis as a Petri net in order to ensure data reproducibility. MaiML files can be converted to PNML files by using XSLT conversion.

2.1.6.2 XES Format

Below <eventLog> elements, some of the elements and attributes defined by XES can be used to ensure measurement/analysis traceability based on event log. MaiML files can be converted to XES files by using XSLT conversion.

2.1.6.3 External Files and Formats

Data formats defined for specific instruments and shared image formats can be used in MaiML files. The <insertion> element can be used to reference files with other formats as external files.

2.1.7 Ensuring Data Tampering Prevention and Confidentiality

MaiML files include tags intended to prevent that file from being modified.

Apply an XML signature (described below) to the entire data file for the MaiML file itself (hereinafter “corresponding MaiML file”). The results include a hash value for the corresponding MaiML file. The purpose is to prevent the corresponding MaiML file from being tampered with.

XMLDSig:Enveloped,　http://www.w3.org/2000/09/xmldsig#enveloped-signature

This includes the XML signature hash value contained in the parent file of the corresponding MaiML file (the unmodified MaiML file on which the corresponding MaiML file is based). It supports block chain guarantees that data tampering is prevented.

For external files, a hash value is generated from the entire file, that hash value is indicated, and the relative address (ULT) of the external file is indicated. The purpose is to prevent tampering with the external file.

Information that requires confidentiality in respective measurement/analysis instruments can be encrypted in MaiML files using partial XML element encryption technology.

XML Encryption: https://www.w3.org/Encryption/2001/

2.1.8 Additional Information

It is recommended that MaiML files should contain the following tags.

2.1.8.1 <uncertainty> Element

The <uncertainty> tag is defined as indicating the uncertainty of data.

2.2 MaiML File Use Cases

Currently, the following two use cases are anticipated for using MaiML to create files.

Case-1: Indicating all steps involved in using instruments and software to perform a series of measurement/analysis processes

In MaiML files, a maimlRootType value is specified as the xsi:type attribute in <maiml> elements.

Case-2: Indicating only the measurement/analysis technique

Indicating only the measurement method without including any data internally

In MaiML files, a protocolFileRootType value is specified as the xsi:type attribute in <maiml> elements.

3. Elements Used in MaiML and How to Indicate Them

3.1 XML Expressions

XML code is text with a hierarchical structure tagged with elements. Elements have names (element names) and the portion enclosed between opening and closing tags, including the element name, is referred to as content. Content can include either text or child elements. Some opening tags can include attributes or attribute values (text).

Example 1

```
<property key="exm1:RoomTemperature">
  <!--opening tag  attribute=attribute value
  The code indicated between the tags is referred to as the content.
  Elements can be included hierarchically.
  closing tag -->
</property>
```

XML expressions in MaiML can be defined in a file referred to as an XML schema. **JIS**-compliant XML schemas are available from JAIMA. XML expressions have a type, just as typical programming languages do. MaiML refers to them as element types. In addition to element types indicated in the main **JIS** standard, the XML schema in MaiML also defines the element types supported by element names used in each hierarchical level of MaiML files.

Note 1 A checker for verifying that XML files conform to MaiML schemas is available from JAIMA.

Note 2 VS Code is a recent editor that includes functionality extensions that can verify whether created files conform to XML and XML schema requirements.

In this context, the concept of a thing or event in typical programming languages is referred to as an object. The object type is referred to as the class, where the class can be thought of as the design drawing for objects. Data that characterizes the class is referred to as class attributes. If actual data is embedded in a class, it is referred to as an instance.

Element names defined by an XML schema in MaiML express the concept of measurement/analysis data (information) or events. Therefore, element names can be thought of as class names. Additionally, because data and metadata in MaiML files are embedded in elements, elements included in MaiML files can be thought of as instances and child elements and element attributes in MaiML elements can be thought of as expressing class attributes.

```
Indicating Elements (Instances)
<element name (class)       attribute (class attribute)>
  Values or Elements (Instances)
</element name>
```

3.2 Meaning of Global Elements

In MaiML files, groups of data obtained by collecting or splitting measurement/analysis data or metadata, such as information about individual samples, conditions, or results, are treated as XML elements (or global elements) with a global element type. Global elements contain a UUID value as the content of <uuid> elements that can be uniquely determined in the measurement/analysis data spaces expressed in MaiML files.

In MaiML, global element names indicate the class name for hierarchically structured data composed of data obtained by collecting or splitting measurement/analysis data or metadata, such as information about individual samples, conditions, or results. In this context, “collecting” refers to collecting data related to the same measurement/analysis but that is distributed among multiple data files or database tables. “Splitting” refers to dividing the data collected according to MaiML class name.

Classes can be characterized by class attributes. In MaiML, class attributes are indicated in terms of child elements or element attributes. In particular, data structures with an unspecified format are indicated in a general purpose data container or a corresponding child element.

Because class attribute values are not defined for MaiML global elements in an XML schema definition, global element names can be considered abstract class names for objects related to highly abstracted measurement/analysis.

On the other hand, vendor and owner organizations can freely collect or split measurement/analysis data. That means they can freely decide the combinations of respective classes used to express how class attributes are used or to express information about conditions, samples, or results. Therefore, global elements include data about the derived abstract MaiML classes defined by vendor or owner organizations.

However, using or combining class attributes differently will result in different classes. In other words, the class namespace is different than the name, so that should be indicated in the <name> element of global elements. In that case, the prefixed XML modifier name (xs:QName) determined by the domain of the vendor or owner organization expresses the class namespace and name.

Note 1 The <name> element is not essential for global elements, so a class name does not necessarily need to be specified. However, if multiple vendor or owner organizations use the same XML modifier names[8] (xs:QName) for identical key attributes of general purpose data containers that are used in combination with other general purpose containers with a completely different type and that are used in a completely different way, then MaiML offers MaiML file users the convenience of being able to differentiate between respective classes by a method other than referencing an element, such as an <instrument>, <vendor>, or <creator> element.

Table 3.1 Child Elements and Attributes for Global Elements Indicated in the Example Above

Element/Attribute	Description	Remarks
<uuid> element	This expresses an identifier used to treat global element instances uniquely within and outside the MaiML file.	Use UUID version 4 unless specified otherwise.
<name> element	The XML modifier name (xs:QName) specified by the vendor or owner organization expresses the class namespace and name.	If no prefix is included, it can be assumed the namespace was determined from the vendor or owner organization domain namespace URL.
<description> element	This indicates a brief description of the class specified by the vendor or owner organization.	It expresses the displayed name or other information, such as used to indicate the class.
<annotation> element	This provides a more detailed description of the class specified by the vendor or owner organization.	
id attribute	This expresses an identifier used to treat global element instances uniquely within MaiML files.	Instances should be defined with a character string and a number of instances that correspond to the class name specified by the vendor or owner organization.
ref attribute	This references the id attributes within MaiML files for closely related global element instances.	Instances are specified for global element types with a reference included.
Other general purpose data container	This is used to indicate data characterized by a class specified by a	

[8]XML modifier names: General purpose data containers are indicated by the XML modifier name “xs:QName” and are composed of a prefix (namespace prefix) and a local name used to indicate an optional namespace.The namespace prefix and the local name are separated by a colon. To specify the prefix portion, the xmlns attribute is used to combine the prefix with a URI (uniform resource identifier) and provide an abbreviated URI.

elements	vendor or owner organization.	

3.3 Using a Petri Net to Indicate a Series of Measurement/Analysis or Other Processes

For manufacturing, testing, inspecting, and other process flows (series of process steps) involved in or related to measurement/analysis (hereinafter “measurement/analysis-related”), it is important to establish a corresponding manual as a protocol and express a history of using the processes in actual operations. In that context, all “measurement/analysis-related” processes have a common characteristic of being able to express measurement/analysis operations and other events as a process flow that corresponds to the flow of time. Measurement/analysis-related process flows can be thought of as a system of discrete events with a constantly changing status composed of a diverse combination of multiple measurement/analysis or other operations.

A Petri net is a model used to represent a system of discrete events[9]. For measurement/analysis-related process flows, Petri nets can be used to model systems as results generated by measurement/analysis instrument and software operation events and the given measurement/analysis samples and condition settings, which correspond to inputs.

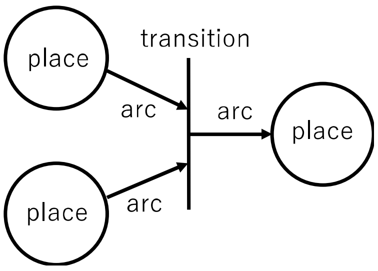
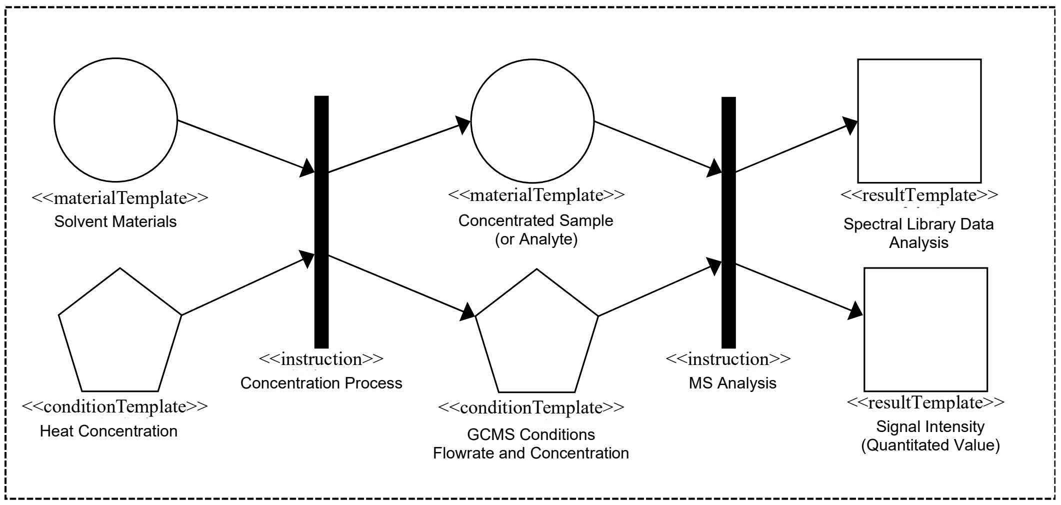


Fig. 3.1 Petri Net

If considering only operations for a specific measurement/analysis method, then input and output values are clearly the metadata and data for samples, condition settings, and so on. However, to indicate processes for multiple operations, including material manufacturing, pretreatment for measurement/analysis, and data analysis post-processing, and also ensure traceability of those processes, then it must be possible to indicate whether respective metadata and data are from input or output values for respective operations.

In the next case, a GC-MS system (gas chromatograph-mass spectrometer) is used for measurements by the GC/MS method (gas chromatography-mass spectrometry).



[9]System of discrete events: Generic term for dynamic systems that transition is discrete steps due to the occurrence of events.

Fig. 3.2 GC-MS Measurement Case

This Petri net shows how the output conditions from pretreatment by evaporative concentration (MS analysis conditions derived from pretreatment) and the materials (pretreated samples) become the inputs for the next MS analysis and connect to output as spectra, library analysis, and signal intensity, and other results.

3.3.1 Indicating Measurement/Analysis Process Flow

For MaiML files, some of the PNML methods for indicating Petri nets are used to indicate a model of the protocol (refer to **Fig. 9.1**).

Normally, transitions due to operations are represented as bars or small squares in Petri nets, but in this case they are indicated as bars, in the same manner as in the main regulation and appendix.

The locations of resources (information) on the input and output sides of transitions are referred to as places, which are represented by circles. In this case, circles are used as a template for sample information, pentagons for condition information, and squares for results information, in the same manner as in the main JIS standard and appendix.

Furthermore, resources (information) on the input and output sides of transitions are referred to as tokens located within places. Tokens are indicated by zero or more black dots or a zero or higher number. In this case, sample information instances, condition information instances, and results information instances are represented by black dots, as they are in the main standard and appendix.

In Petri nets, places on the input side are connected by arcs that represent inputs to the transition (inputs from the places to the transition). Output places are connected by arcs that represent outputs from transitions (outputs from transitions to places). The number of tokens consumed or generated by the firing of each transition is represented by the number of arcs or a number that is 1 or higher. In this case, it is shown as a number or omitted.

Petri net rules specify that transition firings consume a prespecified number (number of arcs on the input side) of input tokens and generate a prespecified number (number of arcs on the output side) of output tokens. The rules also specify that the transition will not fire unless at least the prespecified number of tokens are located at respective input places.

Conversely, that also means that transitions can fire an unlimited number of times as long as N times the prespecified number of tokens are located on respective input places. That not only results in waiting for all transitions for the previous process to fire before the subsequent transition processes can start firing, if firing transitions for the previous process results in the prespecified number of tokens being placed at the input places for the subsequent process, then it also means transitions can be fired for the subsequent process even if transitions have not finished firing for the previous process. Consequently, it is well suited for indicating parallel actions in measurement/analysis or other systems.

In the context of measurement/analysis, “located” can be interpreted as meaning the process is ready to start, “consumed” means the process cannot be repeated, and “generated” means the process is finished. “Cannot repeat process” may be difficult to understand in terms of condition information, but it may be easier to understand if “location” is interpreted as “a link to processes being processed” and “loaded into memory,” whereas “cannot repeat process” after use is easier to understand if interpreted as “discarded from memory.”

Meanwhile, recording parallel actions in a Petri net requires a complicated indication of where tokens are located, consumed, or generated. Therefore, normally parallel actions are indicated using event log rather than tokens when using PNML. MaiML uses a portion of XES indication methods to indicate event log for protocol execution results.

3.3.2 How to Indicate Measurement/Analysis Pretreatment Information

MaiML can also be used to indicate information about pretreatment for measurement/analysis.

Pretreatment protocol outputs can be used not only for <resultTemplate> or <result> elements, but also for <materialTemplate>, <material>, <conditionTemplate>, or <condition> elements. It is expected that outputs from the final operation in the pretreatment process will be used as inputs for one of the operations in the subsequent series of processes.

3.3.3 Precautions for Indicating Integrated Processes for Typical Measurement/Analysis

For typical measurement/analysis process flows, it is expected that the <method> element will be used in corresponding MaiML files to indicate a series of applicable measurement/analysis processes collectively. For measurement/analysis systems, that would mean indicating overall systems as a system of systems. For more details, refer to **4.3.2**.

In this case, if multiple <method> elements are specified as child elements under a <protocol> element, it is anticipated that data from multiple measurement/analysis systems will be used integrated as a single MaiML file. For example, if used for big data analysis, the source for the corresponding big data can be cited as a MaiML file by integrating the files and using an <insertion> or other element to reference the source.

4. Overall File Composition

The following is an overview of language specifications for MaiML files and indicates the intention of respective elements.

<conditionTemplate>, <materialTemplate>, and <resultTemplate> elements can be indicated in multiple hierarchical levels as child elements for <protocol>, <method>, and <program> elements. They are intended for indicating metadata and other information that can be shared for multiple <method> and <program> elements.

> **Note**: The <xxxxTemplate> elements contained in <protocol> elements are used to indicate default values and so on (class or template). They are overwritten by information (instances) indicated by <xxxx> in <data> elements.

4.1 Level 0: <maiml> Element

The entire file is indicated between <maiml> opening and closing tags to indicate a set of data that is independent from other file formats.

4.2 Level 1 Use Cases

For Case-1, level 1 includes four segments with opening and closing tags indicated for <document>, <protocol>, <data>, and <eventLog> elements.

For Case-2, not containing any <data> or <eventLog> elements is permitted. It is based on expressing only design specifications for the entire series of measurement/analysis processes. <data> elements indicate information such as the actual conditions and samples used for measurement/analysis and the results from measurement/analysis operations. The <eventLog> element indicates a log record of measurement/analysis operations, so it is not necessary if only design specifications are indicated.

4.3 Meaning of Respective Elements Used in Level 1 or Below

The corresponding level for the four segments indicated in **4.2** (<document>, <protocol>, <data>, and <eventLog> elements) is indicated in the substructure of each segment.

4.3.1 Level 1: Content of <document> Element

<document> elements (refer to **6.2.1** in the **JIS** standard) include the elements and attributes of global elements (refer to **6.1.4** in the **JIS** standard and **3.2** in this guideline) and also owner, date, and other descendant elements if that MaiML file is created.

<document> elements (refer to **6.2** in the **JIS** standard) are essential because they guarantee the uniqueness of MaiML files and measurement/analysis data. They also include <uuid> elements that guarantee the uniqueness and unambiguousness of <document> elements. Such <uuid> elements are based on random numbers, with unique values contained in each MaiML file.

MaiML files are revised using the <parent> element (refer to **6.2.10** in the **JIS** standard). To indicate uniqueness, UUID values based on random numbers are used for <uuid> element values . Consequently, there is no relation between UUID values in files before (hereinafter “parent” files) and after (hereinafter “child” files) they are revised. Therefore, it cannot be determined whether a file is a revised version or not. To clearly indicate that a file is a revised version, a <parent> element can be used to indicate the relationship between files. The <parent> element indicates the relationship between files by the parent file <uuid> element value and the parent file hash value (<hash> element) contained in the <parent> element.

4.3.2 Level 1: Content of <protocol> Element

In MaiML files, the <protocol> elements indicate the measurement/analysis processes, which makes it essential for data reproducibility. This level is an integration of multiple <method> levels. It is used to specify instructions for measurement/analysis instruments, software, or other purposes when designing measurement/analysis, pretreatment, or other processes.

This can be used to execute a process approach in a quality management systems (**ISO 9001**). It can also be used to indicate measurement/analysis processes, pretreatment processes, or material manufacturing processes when using a PDCA cycle.

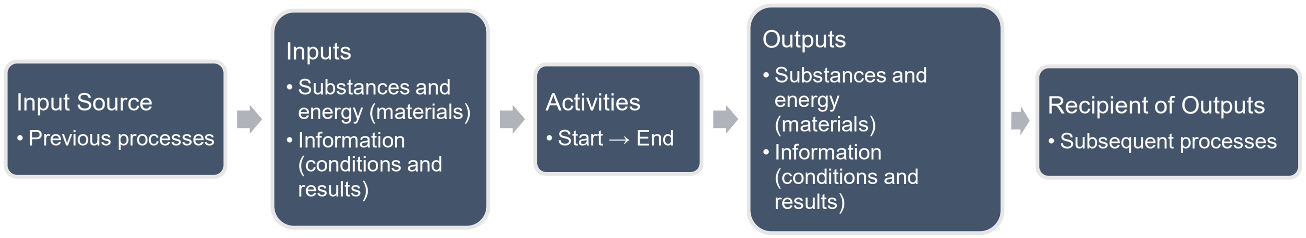

Input Source
• Previous processes
Inputs
• Substances and energy (materials)
• Information (conditions and results)
Activities
• Start → End
Outputs
• Substances and energy (materials)
• Information (conditions and results)
Recipient of Outputs
• Subsequent processes

Fig. 4.1 Diagram of Individual ISO 9001 Processes
Examples of Control and Check Points for Monitoring Performance and Measurements

The process approach uses inputs from previous processes as an input source to generate outputs from starting/ending activities and, assuming subsequent processes exist and as the recipient of outputs, determine individual process elements. In MaiML, process elements are referred to as measurement/analysis processes and the activities in individual process elements are referred to as measurement/analysis operations. As the input source and output recipient, the entire series of previous and subsequent processes, if they exist, are referred to as measurement/analysis processes and can be indicated by connecting the series of individual processes.

In MaiML files, <pnml> elements are used to indicate a series of measurement/analysis processes. Inputs and outputs are indicated using <place> elements and activities (measurement/analysis operations) are indicated <transition> elements.

Furthermore, <place> elements can be connected together as structures uniquely suited for measurement/analysis. They are classified as either <materialTemplate> elements for indicating physical/mechanical sample information, <conditionTemplate> elements for indicating condition information for measurement/analysis parameters, and <resultTemplate> elements for indicating measurement/analysis results.

<protocol> elements can include multiple <method> elements as child elements. Normally, measurement/analysis protocols are expected to contain only one <method> element. Protocols that contain multiple <method> elements are anticipated in cases such as when merging multiple sets of measurement/analysis results into a single MaiML file in cyberspace. Examples of such cases are indicated below.

Example 1: Merging results from multiple measurements/analyses of the same sample into a single MaiML file.
This protocol includes multiple <method> elements and uses a <materialTemplate> element to indicate shared sample information directly below the <protocol> element.

Example 2: Merging multiple measurements/analyses in a single MaiML file in order to compare protocols using the same measurement/analysis instruments.
This uses a <conditionTemplate> element to indicate shared measurement/analysis conditions directly below the <protocol> element. The condition settings specialized for individual protocols are indicated in the <method> element or a child element of the <program> element.

Example 3: Merging multiple measurement/analysis results and using AI to analyze the merged results for subsequent processes.
Input data information used in the AI data analysis is indicated in the <resultTemplate> element and the <result> element derived from the <resultTemplate> element. Consequently, the links to the source data for data analysis can be indicated within MaiML files.

4.3.2.1 Level 2: Content of <method> Element

This level is used to collectively indicate input, process, and output information for a series of measurement/analysis processes (refer to **4.3.2.2**). One or multiple <program> and <pnml> elements are intended between the opening and closing tags in the <method> element. The <method> element is anticipated for use in merging individual measurements/analyses to perform combined measurements/analyses.

<method> elements correspond to <results> elements, which are child elements of <data> elements. Therefore, for collective classes (templates) indicated in a <method> element for a series of measurement/analysis processes, the class is indicated in a <results> element each time an instance element (<material>, <condition>, or <result> element) for templates (<materialTemplate>, <conditionTemplate>, or <resultTemplate> elements) involved in inputs/outputs for multiple <program> elements is executed. If processes are consolidated multiple times, each is specified in a different <results> element.

In that case, <materialTemplate>, <conditionTemplate>, and <resultTemplate> elements for shared use may be indicated in multiple <program> elements. Examples of such cases are indicated below.

Example 1: Performing SEM or TEM measurements with a <materialTemplate> element that indicates the same sample as an input <place> element.

Example 2: When using SEM for imaging process A and an image processing program for image process B, and using a <resultTemplate> element for imaging outputs from process A and a <resultTemplate> element for the input image for process B.

4.3.2.2 Level 3: Content of <program> Element

The <program> element is used to indicate a series of measurement/analysis processes. In this case, the series of measurement/analysis processes refers to processes performed by a single measurement/analysis system and indicated in <pnml> elements (**4.3.2.3**). The system could consist of a single measurement/analysis operation (a <transition> element with an <instruction> element referencing the transition) or the system could consist of multiple measurement/analysis operations.

Specifically, it includes <instruction>, <conditionTemplate>, <materialTemplate>, and <resultTemplate> elements.

Typically, for a specific measurement/analysis, a relational database is created as a data warehouse. In such cases, the key attribute for general purpose data containers that contain an <instruction>, <conditionTemplate>, <materialTemplate>, or <resultTemplate> elements element can be thought of as the key, and child <value> element as the value. Then the general purpose data container values in each template can be replaced with actual values contained in <results> elements for each measurement/analysis.

In addition, individual measurements/analyses can be expressed with a primary key by either indicating the key in a <results> element as one instance of a general purpose data container specified in a <conditionTemplate> element or in a child element of a <materialTemplate> element, or by using a <uuid> element value from a <results> element for an instance from a <program> element as the key.

4.3.2.3 Level 3: <pnml> Element

<pnml> elements are used to express measurement/analysis processes. For more details about Petri nets, refer to **3.3** in this guideline.

Currently, of the elements specified by PNML, only the <place>, <transition>, and <arc> elements are used. A Petri net is a method of notation used to indicate the behavior of discrete event systems and now includes a wide variety of extensions. In particular, <token> elements can be used to simulate system actions or other purposes. For example, if multiple samples are used, they can be used when measurement/analysis operations start, but <token> elements are not used in this case. Conditions for starting automatic measurement/analysis operations and other such information must be indicated in a template element referenced by a <place> element.

A networked database, rather than a relational database, should be used to express measurement/analysis processes.

4.3.2.4 Level 4: <instruction> Element

Individual measurement/analysis operations are expressed using <instruction> elements.

In this case, the granularity of operations is not specified, but it must be possible to specify the start and end points in terms of time values in order to function as a single system.

In particular, MaiML files require indicating an end point for <instruction> elements as an <event> descendant element for <eventLog> elements. The purpose is to ensure measurement/analysis traceability. For information about the XES format used for <eventLog> elements, refer to **2.1.6.2** in this guideline and **B.8** in the **JIS** standard.

4.3.2.5 Levels 2, 3, and 4: Content in <materialTemplate> Elements

General purpose data containers are used to indicate templates for sample information in measurement/analysis operations. It can be used to specify information relevant to measurements/analyses. It is also sometimes used to indicate input data for analytical programs.

This can be replaced with the values in <material> elements indicated in <results> elements corresponding to respective <method> elements. In this case, it is expected that information that should be indicated as sample information in <materialTemplate> elements, specific values for respective samples are indicated in <material> elements as <value> elements of the general purpose data containers with identical key attributes.

<materialTemplate> elements can also be used for outputs from measurement/analysis operations. For example, they can be used for outputs from measurement/analysis pretreatment processes.

4.3.2.6 Levels 2, 3, and 4: Content of <conditionTemplate> Elements

<conditionTemplate> elements are used to indicate condition information for measurement/analysis operations using general purpose data containers. They correspond to measurement/analysis parameters. Condition information is indicated as a template using a <materialTemplate> element. If it is expected that the same measurement/analysis processes will be repeated for each sample specified to be measured/analyzed, as indicated in <material> elements, the condition information will be identical.

This can be replaced with the values in <condition> elements indicated in <results> elements corresponding to respective <method> elements. In this case, the expected method of use is to first specify parameters for the designed measurements/analyses and specify default key attributes and values in <conditionTemplate> elements Then if different than the default values for each measurement/analysis, a general purpose data container with the same key attributes is indicated in a <condition> element and the values during respective measurement/analysis processes are indicated in the corresponding <value> child elements. Examples of such changes are indicated below.

Example 1: Indicating room temperature or other external environmental conditions in the examination room.

The expected room temperature is 20.00 °C, which is indicated in a <conditionTemplate> element. Then the room temperature changes with each measurement, corresponding to the value indicated in the <condition> element, if indicated.

Example 2: Indicating SEM observation points to be measured.

To acquire images of the same field of view, the field of view positions are expected to be indicated in <conditionTemplate> elements. In addition, if the actual positions differ due to instrument precision or other factors, those values are indicated in <condition> elements.

Normally, if multiple fields of view are to be observed, the observation locations are indicated in <condition> elements. In that case, <conditionTemplate> elements are used to indicate field of view positions explicitly. In addition, the acquisition position for each image is also indicated in <condition> elements.

<conditionTemplate> elements can also be as outputs. For example, results from measurement/analysis operations can be used as conditions for measurement/analysis operations in subsequent processes.

4.3.2.7 Levels 2, 3, and 4: Content of <resultTemplate> Elements

The type and meaning of data obtained in measurement/analysis results can be indicated by indicating the general purpose data containers that should be included in the results data as descendant elements of <resultTemplate> elements. Then that can be replaced with the values in <result> elements indicated in

<results> elements corresponding to respective <method> elements.

This element can also be used as inputs for operations in analyses corresponding to subsequent processes. The person designing or indicating the series of measurement/analysis processes can select whether to use <resultTemplate>, <materialTemplate>, or <condtionTemplate> elements in the series of processes. That means any of the elements can be indicated. <materialTemplate> or <condtionTemplate> elements are used if they are expected to be used as inputs for different measurement/analysis operations.

4.3.3 Level 1: <data> Element

The purpose of data levels is for saving measurement/analysis results, conditions, and sample information when actual measurements/analyses are performed. That assumes condition, sample, and results information can be obtained in accordance with the given <protocol> level.

This level might not exist, but that would only occur if only the measurement/analysis protocol was saved, as indicated in Case-2. In the <protocol> level, it is expected to be used as a reference file.

Multiple <results> elements containing sample, condition, and results information related to each operation in multiple measurements/analyses performed in accordance with the protocol indicated in <method> elements are expected to be indicated under <data> elements.

4.3.3.1 Level 2: <results> Element

Within the <data> level structure, <results> elements correspond to <method> elements that are child elements of <protocol> elements described in section **3.3**. <program> and <pnml> elements that are child elements of <method> elements must be used to indicate structures related to the inputs/outputs for the series of measurement/analysis processes. The respective <material>, <condition>, and <result> elements indicated in <results> elements must all be linked using ref attributes to id attributes contained in <materialTemplate>, <conditionTemplate>, or <resultTemplate> templates indicated in descendant elements of <protocol> elements.

<results> elements are expected to only list information for <method> elements indicated as a collective series of measurement/analysis processes that correspond to <place> elements.

The elements included between the opening and closing tags in <results> elements are described in sections **4.3.3.2** to **4.3.3.4**.

4.3.3.2 Level 3: <condition> Element

This element is used to ensure the reproducibility of measurement/analysis, with ref attributes used to connect it to <conditionTemplate> elements and to the process flow specified by PNML. It must contain a UUID value for ensuring the uniqueness of conditions.

URI values can be used to reference external files, such as configuration files required for each measurement/analysis, for example.

In order to ensure <condition> elements used in other MaiML files are used as the identical element in the current MaiML file, the elements must have the same UUID values. Applicable MaiML files must be specified using URI values. If the connection between UUID/URI values is not ensured, then they must be corrected later. If files are converted using a converter, then the UUID values must be entered to ensure elements have the same values.

URI conventions may differ depending on the database or local disc drive involved. By verifying that electronic signature hash (digest) and UUID values are identical, they can be connected later.

4.3.3.3 <material> Element Expressions

This element is used to express sample information, analysis input data, or other information for measurements/analyses, with ref attributes used to connect to <materialTemplate> elements or thereby connect to process flows specified by PNML. They must contain a UUID value for ensuring the uniqueness of samples.

This element is expected to be used as inputs for measurement/analysis operations. But, in Cases-A and -B they can be used as pretreatment for measurement/analysis. When specifying a material or other manufactured item, they can be used as outputs.

Different <material> elements within the same file can be cited by using an <instanceRef> element to reference.

The URI value in an <insertion> element can be used to reference external files, such as other databases or CSV files, for example.

In order to ensure <material> elements used in other MaiML files are used as the identical element in the current MaiML file, the elements must have the same UUID values.

In this case as well, if files are converted using a converter, then the UUID values must be entered to have the same values.

URI conventions may differ depending on the database or local disc drive involved. By verifying that electronic signature hash (digest) and UUID values are identical, they can be connected later.

4.3.3.4 <result> Element Expressions

This element is used to indicate measurement/analysis results, with ref attributes used to connect to <resultTemplate> elements or thereby connect to a series of measurement/analysis processes indicated using <pnml> elements. It must contain a UUID value for ensuring the uniqueness of conditions.

<insertion> or other elements can be used to reference external files using URI values.

4.3.4 Level 1: <eventLog> Element

<eventLog> elements conform to the XES standard format and consist of substructures of three levels of <log>, <trace>, and <event> elements. <eventLog> elements are used to record an event log of measurement and measurement-related actions in order to ensure the traceability of measurements/analyses and data. They can be used to record a log of measurement/analysis ending times or other events.

<eventLog> elements are used to ensure traceability based on the time when data was generated or other information. Because they are linked to <data> elements, they are not indicated if no <data> elements are indicated.

This level is an integration of multiple <trace> levels.

4.3.4.1 Level 2: <log> Element

<log> elements are used to collectively indicate logs from a series of measurement/analysis processes performed multiple times.

4.3.4.2 Level 3: <trace> Element

This level is used to indicate logs from performing a series of measurement/analysis processes one time.

4.3.4.3 Level 4: <event> Element

This is used to indicate a log record of individual operations defined in an <instruction> element for a <program> element for one series of measurement/analysis processes.

5. General Purpose Data Container

In the MaiML data format, general purpose data containers are elements used to indicate data and metadata for measurement/analysis processes. General purpose data containers are used as a collection (set) of data or metadata indicated as content in global elements used to express sample information, conditions, results, or other measurement/analysis information.

In the MaiML data format, general purpose data containers are used to store individual input/output values, names, or other measurement/analysis information as a set of data that includes the data type (name) and content (values). Therefore, general purpose data containers can be used as an associative array with a key attribute as a key.

Table 5.1 Overview of General Purpose Data Containers

General Purpose Data Containers	MaiML Definition	XML Type
Type	xsi:type	attribute
Type (Name)	key	attribute
Value	<value>	Content element

The MaiML data format defines a variety of data types, with values expressed by character strings that are translated based on the specified data type so that data can be analyzed by a computer. That enables the data type (name) to be specified by unique expressions that include a namespace.

The following three elements express general purpose data containers.

1. <property> Element (section **5.1.1**)

   This element mainly contains a single value or set of values.

2. <content> Element (section **5.1.2**)

   This element mainly expresses an axis or other relationship between sets of data.

3. <uncertainty> Element (section **5.1.3**)

   Element used to express uncertainty levels.

An example of indicating the laboratory room temperature with a namespace included is provided below.

Example 1: Indicating that the average laboratory room temperature is 22.5 °C, with a standard deviation of 1.2 °C.

```
<property xsi:type=”xs:double”  key=”exm1:MeanOfLaboRoomTemperature” formatString=”0.0”
units=”centigrade”>
  <value> 22.5 </value>
  <uncertainty xsi:type=”xs:double”  key=”exm1:SDofLaboRoomTemperature” formatString=”0.0”
units=” centigrade”>
    <value> 1.2 </value>
  </uncertainty>
</property>
```

5.1 <property> Element, <content> Element, and <uncertainty> Element

There are three types of general purpose data container.

1. This propertyListType contains content as a general purpose data container, but does not include any value itself.

   It indicates the framework for the data structure.
2. This type corresponds to a <property> element that contains a value.
3. This type corresponds to a <content> element.

For measurement/analysis data and metadata classes, structures, databases, arrays or other general data structures, or to indicate the hierarchy of the framework for data structures, including the relationships between table data, multidimensional vector data, or other data, the propertyListType indicated in **JIS K 0200**, article **7.1.3**, **Table 24** is used as a <property> element type that does not contain a <value> element. (Refer to **5.1.1**, **Example 1**)

To specify data parameters for general data structures in measurement/analysis data, such as classes, structures, databases, or arrays, the <property> element types indicated in **JIS K 0200**, article **7.1.3**, **Tables 25 or 26** can be used as a <property> element type that does not contain a <value> element. (Refer to section **5.1.1**.)

To specify data parameters of data structures that include a relationship between the data, such as between rows and columns of tabular data or between axis data in multidimensional vector data, the <content> element types indicated in **JIS K 0200**, article **7.1.3**, **Table 26** can be used as a <content> element that contains a <value> element. (Refer to section **5.1.2**.)

The <uncertainty> element is used as content for indicating data about the uncertainty of values in <property> parent elements or <content> elements. All <property> or <content> element types listed in **JIS K 0200**, section **7.1.3**, **tables 24** to **26** may be used for <uncertainty> elements (section **5.1.3**).

General purpose data containers can be used as an associative array with a key attribute as a key.

However, because more than one element might include the same key attribute, they must include sibling elements to ensure they are treated as element arrays, rather than single elements. The reason for that is described below.

MaiML does not specify that key attributes for general purpose data containers contained in ordered lists must be unique. Therefore, elements can have sibling elements with the same key attribute. In other words, MaiML permits a list of <property>, <content>, or <uncertainty> elements with a key attribute as a key to not be treated as a simple associative array (dictionary) with single general purpose data container elements as values.

On the other hand, MaiML specifies that parent elements for general purpose data containers must include an ordered list of <property>, <content>, or <uncertainty> elements as child elements. That means that the order of elements must not be changed during loading.

Therefore, general purpose data container searches based on matching key attribute values in the parent element level of general purpose data containers will result in an array of general purpose data container elements. However, if the type is a type indicated in **JIS K 0200 Table 26**, then treating the results as a single mergeable list, as in the example described in **JIS K 0200** section **6.1.4**, would not necessarily be appropriate. In such cases, the elements must be used as an array of individual elements. In other words, there are two ways to interpret “an array of elements with the same key attribute.” It was specified to increase the general applicability of data structures.

Note that though <value> elements can be indicated in parallel by dividing them into child elements of a general purpose data container, they can be treated as a single <value> element by merging them.

5.1.1 <property> Element

One use for the <property> element is to use a propertyListType listed in **JIS K 0200**, section **7.1.3**, **Table 24** that does not include a <value> element to explicitly indicate the class, structures, databases, arrays, or other general data structures of the original measurement/analysis data, or explicitly indicate the framework for data structures, including the relationships between tabular data, multidimensional vector data, or other data. However, the framework for data structures can also be indicated using sample, condition, results, or other global elements for measurements/analyses.

Another use for the <property> element is to specify data parameters in general data structures, such as the classes, structures, databases, or arrays of measurement/analysis data using <property> element types listed in **JIS K 0200**, section **7.1.3**, **Tables 25** or **26**.

Example 1: This example specifies the class (framework) and class attribute (data parameter) for sample information using <property> element types listed in **JIS K 0200**, section **7.1.3**, **Tables 24** or **25**. In this case, “Unknown Sample,” the value for the sample type (SampleType) and “0,” the value for the sample type number (SampleTypeNumber) indicate that the sample is a unknown sample.

```
<property xsi:type="propertyListType" key="ns1:SampleInformation">
  <property xsi:type="stringType" key="ns1:SampleId">
    <value>S123-00001-001#001</value>
  </property>
  <property xsi:type="stringType" key="ns1:SampleName">
    <value> Specimen for testing content of impurity XXX</value>
  </property>
  <property xsi:type="stringType" key="ns1:SampleType">
    <value>Unknown Sample</value>
  </property>
  <property xsi:type="unsignedByteType" key="ns1:SampleTypeNumber">
    <value>0</value>
  </property>
</property>
```

Example 2: This example specifies candidate combinations of sample type (SampleType) and sample type number (SampleTypeNumber) values (data parameters) using multiple <property> elements with the same key attribute key as listed in **JIS K 0200**, section **7.1.3**, **Table 25**. It enables multiple arrays to be included in a single ordered list (framework) so that key attribute <property> elements appear in the same order as respective arrays and can also be defined so that sample type (SampleType) values correspond to sample type number (SampleTypeNumber) values listed in the same order.

```
<property xsi:type="propertyListType" key="ns1:SampleTypeEnumerators">
  <property xsi:type="stringType" key="ns1:SampleType">
    <value>Unknown Sample</value>
  </property>
  <property xsi:type="unsignedByteType" key="ns1:SampleTypeNumber">
    <value>0</value>
  </property>
  <property xsi:type="stringType" key="ns1:SampleType">
    <value>Standard Sample</value>
  </property>
  <property xsi:type="unsignedByteType" key="ns1:SampleTypeNumber">
```

```
    <value>1</value>
  </property>
</property>
```

Example 3: This example specifies candidate combinations of sample type (SampleType) and sample type number (SampleTypeNumber) values (data parameters) using <property> elements listed in **JIS K 0200**, section **7.1.3**, **Table 26**. In this example, candidate sample type (SampleType) values are listed based on the stringEnumType, which is the <property> element type that can include blank characters, and based on the unsignedByteListType element type, which is the <property> element type that uses candidate sample type number (SampleTypeNumber) values as the type. In this case, the unsignedByteListType <value> elements contain space-delimited lists of xs:unsignedByte values as content, but the content is split into multiple <value> elements that each contain one list that corresponds to stringEnumType <value> elements. In this example, merging the <value> elements would express the same meaning.

```
<property xsi:type="propertyListType" key="ns1:SampleTypeEnumerators">
  <property xsi:type="stringEnumType" key="ns1:SampleTypes">
    <value>Unknown Sample</value>
    <value>Standard Sample</value>
  </property>
  <property xsi:type="unsignedByteListType" key="ns1:SampleTypeNumbers">
    <value>0</value>
    <value>1</value>
  <!-- <value> 0 1 </value> would indicate the same meaning. -->
  </property>
</property>
```

Example 4: A single array can also be divided between two or more <property> elements, as indicated in the example in **JIS K 0200**, section **6.1.4**. However, it would be interpreted as either one array divided between two <property> elements or as an array of an array. Moreover, because it is not defined in the **JIS** standard, the interpretation cannot be decided based on the **JIS** standard alone. In such cases, some sort of method for making a decision must be indicated separately. The following shows an example of dividing a single array between two or more <property> elements, as described in the example in **JIS K 0200**, section **6.1.4**.

```
<property xsi:type="intListType" key="ns1:SampleList">
    <description>List Item in a Split Sample List</description>
    <value>1 2 3</value>
</property>
<property xsi:type="intListType" key="ns1:SampleList">
    <description>List Item in a Split Sample List</description>  <value>4 5 6</value>
</property>
```

The following shows an example of an array of array lists divided between two or more <property> elements.

```
<property xsi:type="intListType" key="ns1:SampleList">
    <description>List Item in the List of Sample Lists</description>
    <value>1 2 3</value>
</property>
<property xsi:type="intListType" key="ns1:SampleList">
    <description>List Item in the List of Sample Lists</description>  <value>4 5 6</value>
</property>
```

5.1.2 <content> Element

<content> elements can be used to specify row or column data in tabulated measurement/analysis data or to specify data in data structures that include relationships between different data, such as a multidimensional vector data series or axis data, based on <content> element types listed in **JIS K 0200**, section **7.1.3**, **Table 26**.

Note 1 They are used in cases such as to indicate the content of CSV files that include a header row using MaiML general purpose data containers. Content can also be indicated as an external file using an <insertion> element.

In this case, the list type or enumeration type for <property> elements indicated in **JIS K 0200**, section **7.1.3**, **Table 26** can be used in combination with <property> elements listed in **Table 25** to independently explicitly indicate setting values or candidate setting values if the number of list items is fixed or setting values are too low. This element is used if the number of items in a list is limited and their relative meanings can be easily interpreted.

In contrast, the <content> element in **Table 26** is expected to be used for expressing tabular data or multidimensional vector data with a variable number of list items. Consequently, it is used to specify the number of list items with a size attribute when listing multiple <content> elements or specify their hierarchy. In addition, <content> elements can be used to specify table rows or columns or to specify axis attributes for axes in multidimensional vector data.

If <content> elements with the same size attribute are listed as content in <property> elements that include “propertyListType” as the xsi:type attribute value, they can be thought of as expressing a table. Listed <content> elements express respective vector type data arrays, which are permitted to express content for either table columns or rows.

Generally, abbreviated names are specified as axis attributes for rows and columns in tabular data or for series and axis in multidimensional vector data, with full-length names specified for <description> elements. In such cases, it is necessary to beware that the axis attribute meaning is simply the abbreviated name and that it defines the position of data within parent data structures.

Columns and rows in tabular data are respectively either a list of <content> elements containing record indicators and/or names or a list of record data listed successively with <content> elements for each data item. The number of data values included in each data item is equivalent to the number of records (number of list items) and can be indicated explicitly with size attributes.

Example 1: In the following example, row numbers in tabular data indicate record numbers, whereas optional record names and required X, Y, and intensity values are used to indicate column data based on <property> elements indicated in **Table 24** and <content> element types indicated in **Table 26** of **JIS K 0200**, section **7.1.3**. Note that <content> elements contain multiple data values, so the name specified as the key attribute is singular.

In this example, table columns are expressed with a single <content> element. Note that it is different than the method of storing values used for the CSV format. The number of data values is specified by the size attribute.

Index	Recode Name	X (m)	Y (m)	Intensity (V)
1	(Blank)	10.5.	100.2	1.0001E-30
2	Peak Data Point #1	20.2	200.5	2.3204E03
3	(Blank)	30.7.	300.3	1.0011E-26

```
<property xsi:type="propertyListType" key="ns1:MeasurementRecodeResultTable">
  <content xsi:type=" contentUnsignedLongListType" key="ns1:Index" axis="Index" size="4">
```

```
    <description>Recode Index</description>
    <value>1 2 3</value>
  </content>
  <content xsi:type="contentStringEnumType" key="ns1:Name" axis="Recode Name" size="3">
    <description>User Defined Recode Name</description>
    <value></value>
    <value>Peak Data Point #1</value>
    <value></value>
  </content>
  <content xsi:type="contentDoubleListType" key="ns1:X" axis="X" size="3" formatString="0.0"
units="m">
    <description>X-axis Value</description>
    <value>10.5 20.2 30.7</value>
  </content>
  <content xsi:type="contentDoubleListType" key="ns1:Y" axis="Y" size="3" formatString="0.0"
units="m">
    <description>Y-axis Value</description>
    <value>100.2 200.5 300.3</value>
  </content>
  <content xsi:type="contentDoubleListType" key="ns1:Intensity" axis="Intensity" size="3"
formatString="0.0000E00" units="V">
    <description>Measured Intensity</description>
    <value>1.0001E-30 2.3204E03 1.0011E-26</value>
  </content>
</property>
```

Example 2: This example shows the same information as **Example 1** but stored in a format similar to the CSV format. In this case, table rows are expressed with a single <content> element. The number of fields is specified by the size attribute. Also note that respective values are expressed as character strings because a data type cannot be assigned to values.

```
<property xsi:type="propertyListType" key="ns1:MeasurementRecodeResultTable">
  <content xsi:type=" contentStringEnumType" key="ns1:Header" axis="Header" size="5">
    <description>Header</description>
    <value>Index</value>
    <value>Record Name</value>
    <value>X(m)</value>
    <value>Y(m)</value>
    <value>Intensity(V)</value>
  </content>
  <content xsi:type="contentStringEnumType" key="ns1:Record01" size="5">
    <description> Recode 01</description>
    <value>1</value>
    <value></value>
    <value>10.5</value>
    <value>100.2</value>
    <value>1.0001E-30</value>
  </content>
   <content xsi:type="contentStringEnumType" key="ns1:Record01" size="5">
    <description> Recode 01</description>
    <value>2</value>
    <value>Peak Data Point #1</value>
    <value>20.2</value>
    <value>200.5</value>
    <value>2.3204E03</value>
  </content>
  <content xsi:type="contentStringEnumType" key="ns1:Record01" size="5">
```

```
    <description> Recode 01</description>
    <value>3</value>
    <value></value>
    <value>30.7</value>
    <value>300.3</value>
    <value>1.0011E-26</value>
  </content>
</property>
```

Multidimensional vector data is listed if a <content> element with serial index and/or serial name data is available. Then that is followed by listing <content> elements with independent axes not dependent on other axes. For time resolution measurements, it is recommended that <content> elements with serial index and/or time axis information be listed first, followed by listing <content> elements with information for other independent axes. For axes dependent on an independent axis, such as any combination of items in X-index and/or X-axis lists (with x number of list items) or Y-index and/or Y-axis lists (with y number of list items), if there is an intensity axis, then <content> elements for X and Y axes are listed followed by <content> elements for the intensity axis (with x times y number of list items).

Example 3: Example with an **intensity** axis dependent on X and Y axes.

```
<property xsi:type="propertyListType" key="ns1:MeasurementMatrix">
  <content xsi:type="contentDoubleListType" key="ns1:X" axis="X" size="3"
formatString="0.0" units="m">
    <description>X-axis Value</description>
    <value>10.5 20.2 30.7</value>
  </content>
  <content xsi:type="contentDoubleListType" key="ns1:Y" axis="Y" size="3"
formatString="0.0" units="m">
    <description>Y-axis Value</description>
    <value>100.2 200.5 300.3</value>
  </content>
  <content xsi:type="contentDoubleListType" key="ns1:Intensity" axis="Intensity"
size="9" formatString="0.0000E00" units="V">
    <description>Measured Intensity</description>
    <value>4.0001E-30 2.3204E03 1.0011E-26</value><!-- X=10.5,   Y=100.2 200.5
300.3 -->
    <value>1.0010E-23 1.2045E02 9.0401E-30</value><!-- X=20.2,   Y=100.2 200.5
300.3 -->
    <value>8.0111E-33 4.5278E04 1.5018E-20</value><!-- X=30.7,   Y=100.2 200.5
300.3 -->
  </content>
</property>
```

5.1.3 <uncertainty> Element

<uncertainty> elements can contain either <property> or <content> element types specified in **JIS K 0200**, section **7.1.3**, **Tables 24** to **26**, with the element type determined by the <property> or <content> elements in ancestor or sibling elements. However, those element types determine the uncertainty of <property> and <content> elements and are not necessarily the same as ancestor or sibling <property> or <content> element types.

For example, in regions with unknown time zones or with Coordinated Universal Time (UTC), date and time (xs:dateTime) values or corresponding lists, the <property> element for the dataTimeType type indicated in **Table 25** of **JIS K 0200**, section **7.1.3**, or the dateTimeListType type indicated in **Table 26**, or the <content>

element for the contentDateTimeListType type indicated in **Table 25** might include an <uncertainty> element as an offset for the local time of the owner organization, but the element types in **Table 25** that can store a time zone offset value are stringType and tokenType types.

5.2 XSD Data Types

5.2.1 xs:dateTime Type

The xs:dateTime type is a date/time type used for <date> elements, dateTimeType <value> elements listed in **JIS K 0200**, section **7.1.3**, **Table 25**, as potentially for <property> or <uncertainty> elements, or for space-delimited list items in dateTimeListType type <value> elements that potentially could be assigned <property> or <uncertainty> element types listed in **Table 26**, or for space-delimited list items in contentDateTimeListType <value> elements that potentially could be assigned <content> or <uncertainty> element types listed in **Table 26**.

Of the possible formats for indicating dates and times, the format for xs:dateTime types is defined as the **ISO 8601** format for indicating dates and times in combination and that satisfies **JIS X 0301**, section **5.4.1 a)** requirements for indicating calendar days.

YYYY-MM-DDThh:mm:ss.sssTZD

In this case, “YYYY,” “MM,” and “DD” refer to the number of digits in the year, month, and day values in calendar dates. The “T” symbol expresses the time value, where ”hh,” “mm,” and “ss” values indicate the number of digits in hour, minute, and second values and where no delimiting characters or digits can be omitted. A decimal point is only permitted for second values. Decimal values can be specified by indicating the decimal point and the desired number of decimal places with “s” characters.

In addition, the “TZD” specifies displaying Coordinated Universal Time (UTC) values, as specified in **JIS X 0301** (**ISO 8601**) and indicated with a “Z” symbol. “±hh:mm” is displayed to indicate the time difference between the local and UTC times (time zone offset) is displayed, whereas “±hh:mm” is not displayed if the local time is displayed.

The correct date and time format is shown below.

Correct Date/Time	Remarks
2022-02-05T09:00:00	This shows the local with an unknown time zone offset. Though the geographical information (longitude) is vague, it includes information about the time of day. However, even if the time zone offset cannot be determined in event log or other contexts, the converter creator should consider adding a note to <uncertainty> elements about compensating for the time zone offset of the data owner location.
2022-02-05T09:00:00Z	This shows the UTC time with an unknown time zone offset. This is suitable for exchanging data internationally, but it does not include geographical (longitudinal) or time-of-day information. However, even if the time zone offset cannot be determined in event log or other contexts, the converter creator should consider adding a note to <uncertainty> elements about compensating for the time zone offset of the data owner location.
2022-02-05T00:00:00+09:00	This shows the local time with the specified time zone offset. It is suitable for exchanging data internationally and also includes geographical (longitudinal) and time-of-day information.
2022-02-05T09:00:00.000	This format shows values less than one second in millisecond units. This is the most commonly used time precision level used for computer system log.
2022-02-05T09:00:00.000000	This format shows values less than one second in microsecond or 100 nanosecond units.

2022-02-05T09:00:00.00000000	However, some processing systems (depending on the operating system or program language) may not be able to process the values. Therefore, the person creating the converter should include a note about values less than 1 millisecond in a <property> element.

The following date and time formats are invalid.

Invalid Date/Time Values	Remarks
2022	The integer portion of year, month, date, hour, minute, and second values are required and cannot be omitted.
2022-02	
2022-02-05	
2022-02-05T09	
2022-02-05T09:00	
02-05T09:00:00	
22-02-05T09:00:00	All digits in the integer portion of year, month, date, hour, minute, and second values are required and cannot be omitted.
2022-2-5T9:00:00	
28-02-2022T09:00:00	Dates must be indicated in the order of year, month, and date.
2022/11/28 12:00:00	The delimiters between year, month, and day values in calendar dates, the T time symbol, and the delimiters between hour, minute, and second values are required and cannot be omitted or changed.
20221128 120000	
2022-02-05T09:00:00JST	Standard time names cannot be used.
2022-02-05T09:00:00Z+00:00	The "Z" UTC symbol and the time zone offset value cannot both be displayed.
2022-2-5T9:00:00+9:00	None of the digits for time zone offset hour and minute values may be omitted.
2022-02-05T09:00:00+09	Minute values cannot be omitted from time zone offset values.
2022-02-05T09:00:00+09:00:00	Seconds cannot be specified for time zone offset values.

5.2.2 xs:decimal Type

The xs:decimal type is used for decimalType <value> elements listed for <property> and <uncertainty> elements in **Table 25** of **JIS K 0200**, section **7.1.3**, for items listed in space-delimited lists for decimalListType <value> elements listed for <property> and <uncertainty> elements in **Table 26**, or for base-ten real number items listed in space-delimited lists for contentDecimalListType <value> elements listed for <content> and <uncertainty> elements in Table 26. In the XML schema it is defined as the type for base-ten real numbers with any number of integer places and any number of decimal places.

Unlike the xs:float and xs:double types, the xs:decimal type has various restrictions, such as not permitting scientific notation, infinitely large (±INF) positive or negative values, or non-numeric (NaN) values.

However, because the standard for binary expression of base-ten real numbers is defined as the **ISO/IEC/IEEE 60559**:2011 (IEEE 754-2008) standard for base-ten floating decimal real numbers, which is a relatively new standard, some programming languages will limit the number of digits or cannot comply with the standard in its standard form. Therefore, particular care is required for processing the type in programming languages.

In general, if a programming language supports base-ten real numbers, then base-ten real numbers are expressed in the form

(-1) sign × integer / 10 scale

where real numbers are usually defined based on a set of (sign, integer, and scale) or equivalent values (typically values from an offset scale). In this case, the sign value is either 1 or 0 to indicate whether or not a sign is included, the integer value is 0 or a positive integer with an arbitrary number of digits, and the scale value is 0 or a positive integer with an arbitrary number of digits. If the scale value is 0 or a positive integer, it indicates the number of decimal places. If it is a negative integer, it indicates the number of decimal places to shift base-ten places (to the left). That is the reason base-ten floating decimal real numbers are defined for expressing base-ten real numbers in binary form. The difference from base-two floating decimal real numbers is that it cannot generate an error in the decimal portion.

A key precaution for processing the xs:decimal type as a binary expression of base-ten real numbers using an **ISO/IEC/IEEE 60559**:2011 (IEEE 754-2008)-compliant programming language is the risk of the binary expressions being converted to scientific notation, an infinitely large positive or negative (±INF) value, or non-numeric characters. For scientific notation, make sure scientific notation is not used for character strings. For infinitely large positive/negative or NaN (non-numeric) values, due to the risk of overflows, division by zero, or other results could become embedded in original measurement/analysis data, the converter creator may need to define upper/lower limit values or substitute values for replacing non-numeric values in a <property> element.

Even if the programming language supports binary expression of **ISO/IEC/IEEE 60559**:2011 (IEEE 754-2008)-compliant high-precision base-ten real number types, obtaining and calculating large amounts of base-ten real numbers with an arbitrary number of digits would consume large amounts of memory. In reality, that might require information about the actual range and precision of the xs:decimal type if the maximum integer value was set low or if a maximum integer value needs to be specified. It may be necessary for the converter creator to define information about the actual range or precision for xs:decimal types for values with a large number of digits.

If the programming language does not ansupport binary expression of **ISO/IEC/IEEE 60559**:2011 (IEEE 754-2008)-compliant base-ten real number types, then consider using a base-two floating decimal real number type (double) with several times higher precision, while also being careful of the number of significant digits, the exponent range, and decimal error rounding processes or consider using a third-party library.

If the programming language supports binary expression of base-ten real number types that are not compliant with **ISO/IEC/IEEE 60559**:2011 (IEEE 754-2008), then consider using that type while being careful of the number of significant digits and the exponent range, using a base-two floating decimal real number type (double) with several times higher precision while being careful of the number of significant digits, the exponent range, and decimal error rounding processes, or consider using a third-party library.

5.2.3 xs:float and xs:double Types

xs:float and xs:double types are used for floatType and doubleType <value> elements listed for <property> and <uncertainty> elements in **Table 25** of **JIS K 0200**, section **7.1.3**, for space-delimited items listed in floatListType and doubleListType type <value> elements listed for <property> and <uncertainty> elements in **Table 26**, for space-delimited items listed in contentFloatListType and contentDoubleListType type <value> elements listed for <content> and <uncertainty> elements in **Table 26**, and for scaleFactor types used as a real number type for <property>, <content> or <uncertainty> elements. In the XML schema, they are defined as types for 32 and 64-bit floating decimal real numbers.

Requirements for 32 and 64-bit floating decimal real number types is defined in **ISO/IEC/IEEE 60559** (IEEE 754) as the type for base-two floating decimal real numbers.

Though scientific notation, infinitely large positive and negative (±INF) values, or non-numeric values (NaN) can be displayed for xs:float and xs:double types, in the same manner as for programming language float and double types, but that requires being careful about deviating from the xs:float or xs:double type format when indicating programming language float and double types as character strings and being careful of rounding errors for the decimal portions of <value> element values and scaleFactor attribute values.

Note, however, that the xs:double type is used for the scaleFactor attribute type, because the xs:decimal type does not permit scientific notation.

The converter creator does not guarantee that the product of the <value> element value times the scaleFactor attribute value will be within the range and precision specified for the original <value> element type. Therefore, MaiML file users need to select an appropriate type by also considering programming language-specific problems and then process types properly, including handling rounding errors.

5.2.4 xs:string and List (xs:list) Types

The xs:string type is used for stringType <value> elements listed for <description>, <annotation>, <property>, and <uncertainty> elements in **Table 25** of **JIS K 0200**, section **7.1.3** or stringEnumType <value> elements in **Table 26**, for items listed in space-delimited lists for stringListType <value> elements listed for <property> and <uncertainty> elements in **Table 26**, for items listed in space-delimited lists for contentStringListType <value> elements listed for <content> and <uncertainty> elements in **Table 26**, or for character strings with blank characters retained for contentStringEnumType <value> elements listed for <content> and <uncertainty> elements in **Table 26**.

Blank characters (space, tab, line feed, or carriage return characters), contained in items listed in **JIS K 0200**, section **7.1.3**, **Table 26**, for stringListType and contentStringListType types are processed as list delimiters. Numbered entity reference strings “&#x20;,” “&#x09;,” “&#x0A;,” and “&#x0D;” cannot be used as an escape sequence because they are processed as character strings. In such cases, either avoid using non-XML escape sequences, such as the “\u0020,” “\t,” “\n,” “\r,” or “\\” strings used in the C language as space, tab, line feed, carriage return, or reverse solidus (backslash) escape sequence character strings for printf or use a stringEnumType or contentStringEnumType type indicated in **5.1.1 Example 3**, rather than a stringListType or contentStringListType type.

Example 1 The following is an example of **Example 3** in **5.1.1** rewritten using a stringListType listed in **Table 26** in **JIS K 0200 7.1.3**. In this example, the code would be interpreted as listing four items rather than two.

```
<property xsi:type="stringListType" key="ns1:SampleTypes">
  <value>Unknown Sample</value>
  <value>Standard&#x0020;Sample</value>
</property>
```

Example 2 The following is an example of **Example 3** in **5.1.1** rewritten using a stringListType listed in **Table 26** of **JIS K 0200 7.1.3**. In this example, the following list is correctly interpreted as listing two items, by using the C language space escape sequence (\u0020) for printf as an independently defined space escape sequence to avoid spaces contained in listed items from being interpreted as delimiting characters. However, the method requires canceling the escape sequence whenever the data is used. Therefore, the converter creator has included information for data users in a <property> element with a “ns1:EscapeSequenceInC” key attribute.

```
<property xsi:type="stringListType" key="ns1:EscapeSequenceInC">
  <value>\u0020 \t \n \r \\</value>
</property>
<property xsi:type="stringListType" key="ns1:SampleTypes">
  <value>Unknown\u0020Sample</value>
  <value>Standard\u0020Sample</value>
</property>
```

Spaces or tabs meaningful for indicating carriage returns or indentations in the original measurement/analysis data would become intermingled with spaces and tabs with no meaning for data but used to format XML code. That could cause problems for some XML viewer/editor programs, such as when displaying trees. Such problems can be avoided by using the printf escape sequence or other method to convert blank characters.

Example 3 In this example, a carriage return is included in the sample name indicated in **Example 1** in **5.1.1**. The carriage return is properly retained in the <value> element, but some XML viewer/editor programs will no longer be able to display the levels properly.

```
<property xsi:type="propertyListType" key="ns1:SampleInformation">
  <property xsi:type="stringType" key="ns1:SampleName">
    <value> Specimen for testing content of impurity XXX.
Diluted. </value>
  </property>
</property>
```

Example 4 Example of using the printf escape sequence “\r\n” to escape from a carriage return.

```
<property xsi:type="propertyListType" key="ns1:SampleInformation">
  <property xsi:type="stringType" key="ns1:SampleName">
    <value> Specimen for testing content of impurity XXX. \r\n Diluted. </value>
  </property>
</property>
```

If a blank character is included in a stringListType or contentStringListType type listed in **JIS K 0200**, section **7.1.3**, **Table 26**, the delimiter characters before and after that list item will be compressed, shifting the data position in the list by one position.

Example 5 Example with the fifth and seventh peaks marked with a “V” to indicate the peak shape classification. In this example, the four delimiter characters (blank characters) that indicate unmarked peaks before/after marked peaks and the two intermediate delimiter characters in the list are compressed, so that the list is interpreted as including two items.

```
<content xsi:type="contentStringListType" key="ns1:gcmsPeakTableMark" axis="Mark"
size="11" id="content_gcmsPeakTable_Mark" ref="contentTemplate_gcmsPeakTable_Mark">
  <description>Peak Mark</description>
  <value>    V  V    </value><!-- Interpreted as “V V” -->
</content>
```

In this case, an idRefListType, contentIdRefListType, qualifiedNameRefListType, or contentQualifiedNameRefListType type is used to list data for each item in <property> elements separately, rather than using the stringListType or contentStringListType type indicated in **JIS K 0200** section **7.1.3**, **Table 26**.

Example 6 Example of **Example 5** properly rewritten using idType and contentIdRefListType types.

```
<property xsi:type="propertyListType" key="ns1:peak_marks">
  <property xsi:type="idType" key="peak_mark_key">
    <value>blank_id</value>
    <property xsi:type="stringType" key="ns1:peak_mark_value">
      <value></value>
    </property>
  </property>
  <property xsi:type="idType" key="ns1:peak_mark_key">
    <value>mark_v_id</value>
    <property xsi:type="stringType" key="ns1:peak_mark_value">
      <value>V</value>
    </property>
  </property>
</property>
<content xsi:type="contentIdRefListType" key="ns1:gcmsPeakTableMark" axis="Mark"
```

```
size="11" id="content_gcmsPeakTable_Mark" ref="contentTemplate_gcmsPeakTable_Mark">
  <description>Peak Mark</description>
  <value>blank_id blank_id blank_id blank_id mark_v_id blank_id mark_v_id blank_id blank_id
blank_id blank_id</value>
</content>
```

Example 7 Example of **Example 5** properly rewritten using the contentQualifiedNameRefListType type.

```
<property xsi:type="propertyListType" key="ns1:pre_defined_string_qnames">
  <property xsi:type="stringType" key="ns1:blank">
    <value></value>
  </property>
  <property xsi:type="stringType" key="ns1:mark_v">
    <value>V</value>
  </property>
</property>
<content xsi:type="contentQualifiedNameListType" key="ns1:gcmsPeakTableMark"
axis="Mark" size="11" id="content_gcmsPeakTable_Mark"
ref="contentTemplate_gcmsPeakTable_Mark">
  <description>Peak Mark</description>
  <value>ns1:blank ns1:blank ns1:blank ns1:blank ns1:mark_v ns1:blank ns1:mark_v
ns1:blank ns1:blank ns1:blank ns1:blank </value>
</content>
```

If any of the listed items contain data that might require confidentiality, it is not possible to make only some list type or enumeration type listed items confidential. (Confidentiality can only be applied collectively.) In such cases, use an idRefListType, contentIdRefListType, qualifiedNameRefListType, or contentQualifiedNameRefListType type to list data for each item in <property> elements separately, rather than using the stringListType or contentStringListType type indicated in **JIS K 0200** section **7.1.3**, **Table 26**.

Example 8 Example of peaks that include owner-defined compound names that might need to be kept confidential.In this case, listed items are defined individually using the contentQualifiedNameRefListType type. The “ns1:blank” string indicated in **Example 7** is used for blank characters. If any data other than the blank characters in arbitrary initial character string values was entered in a list, the converter creator should successively create and store <property> elements with a key created based on the compound number or peak number, such as “ns1:peak5_1” or “ns1:peak7_1.” However, if stringType is used as the type, then alternative explanatory comments cannot be added when <value> element is made confidential. Therefore, using “ns1:original_name” as the key provides the freedom to create and store child <property> elements, so that sibling <property> elements can be created for supplementing content after confidentiality is applied.

```
<property xsi:type="propertyListType" key="ns1:pre_defined_string_qnames">
  <property xsi:type="propertyListType" key="ns1:peak5_1">
    <property xsi:type="stringType" key="ns1:original_name">
      <value>Peak #1 of Compound AAA</value>
    </property>
  </property>
  <property xsi:type="propertyListType" key="ns1:peak7_1">
    <property xsi:type="stringType" key="ns1:original_name">
      <value>Peak #1 of Compound XXX</value>
    </property>
  </property>
</property>
```

```
<content xsi:type="contentQualifiedNameListType" key="gcmsPeakTableName" axis="Name"
size="11" id="content_gcmsPeakTable_Name"
ref="contentTemplate_gcmsPeakTable_Name">
  <description>Peak Name</description>
  <value>ns1:blank ns1:blank ns1:blank ns1:blank ns1:peak5_1 ns1:blank ns1:peak7_1
ns1:blank ns1:blank ns1:blank ns1:blank </value>
</content>
```

Example 9 Confidentiality can be assigned individually to listed items referenced in the list in **Example 8** without changing the list. In this case, confidentiality was applied to the content in the child <property> element with the key “ns1:original_name” specified as the key for the <property> element with the key “ns1:peak7_1” specified in **Example 8**. Then the alternate name specified by the owner was assigned to the <property> element with the key “ns1:anonymous_name” as a sibling element when confidentiality was applied.

```
<property xsi:type="propertyListType" key="ns1:pre_defined_string_qnames">
  <property xsi:type="propertyListType" key="ns1:peak5_1">
    <property xsi:type="stringType" key="ns1:original_name">
      <value>Peak #1 of Compound AAA</value>
    </property>
  </property>
  <property xsi:type="propertyListType" key="ns1:peak7_1">
    <property xsi:type="stringType" key="ns1:original_name">
      <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
                           xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
          <CipherData>
            <CipherValue>

hpN5YIKQ2gMgc6rAav9NyCSrRk17XQG1qxAWvBf7NUhMj2nBhVZm9GabnCs7MJC9
             UqTU30OoZBCNsWBfjd+6xA==
            </CipherValue>
          </CipherData>
    </EncryptedData>
    </property>
    <property xsi:type="stringType" key="ns1:anonymous_name">
      <description>Anonymous or Alternative Name of Encrypted Original Name</description>
      <value>Peak #1 of Anonymous Compound</value>
    </property>
  </property>
</property>
```

6. How to Cite External Files

In MaiML files, files located on multiple external devices (either local or on the Internet) (hereinafter “external files”) can be specified by designating them. In this context, “local” refers to ZIP format archived files, whereas “on the Internet” refers to files identified with a URI[10] value.

Reasons for designating external files include the following.

- To handle binary files for high-speed processing or to handle very large files
- To handle some or all measurement/analysis condition settings as vendor-specific files
- To cite the same file from multiple MaiML files

[10]URI (Uniform Resource Identifier): A standard data format defined for uniquely identifying information, services, devices, or other resources.

- To cite a lock, password, or other file on the Internet for restricting access

6.1 How to Cite Files

External files are cited by specifying an <insertion> element. In that case, the external file information included in the <insertion> element as content includes a <uri> element that specifies the file position and a <hash> element that contains the file hash value. Furthermore, if the external file is based on the MaiML data format, the UUID value contained in the <document> element is indicated in the <uuid> element. The <format> element specifies the format of the external file. An example is indicated below.

Example 1 The following is an example of indicating an <insertion> element.

```
<insertion>
  <uri> http://www.example.com/maiml/material#catalogue00</uri>
  <hash method="SHA-256">
    fe69081735a15705a11872621a5b34bc5da50b2626c95ffde234d9d79a3b7432</hash>
  <uuid> 246af27a-1f7c-4f64-817f-cd58d7c7ef13</uuid>
  <format>application/maiml</format>
</insertion>
```

<uri>, <hash>, or <uuid> element

Details are described in the following steps.

(1) Compress Local External Files

To locate an external file locally, it must be converted to the ZIP format. In that case, the file extension “.maiml.zip” is used.

(2) Decide the URI

Specify a URI (RFC3305) that can express either local files in ZIP format or files located on a network (external files indicated above).

Example 1 <uri> file:xxx.xxx</uri> for a local file

Example 2 <uri> http://yyyy/xxx.xxx</uri> for a file on the Internet

(3) Decide the Hash Value

Use a SHA-2 hash function, such as SHA-256 or SHA-512 to generate hash values from the entire external file. Hash values are numbers with a fixed number of digits determined from the original data in external files by a defined calculation procedure referred to as a hash function. The original data cannot be reproduced from hash values. Hash functions are specified using method attributes. An example is described below.

Example 1 The following uses an SHA-256 value to express 256-bit (64-byte) values.

```
<hash method="SHA-256">
312f368d27bd5c9d76bab23a3bf0b82338c2a68377cc2b58012daae132f7be6d
</hash>
```

If even one character is changed, a completely different number is assigned. Therefore, it guarantees the uniqueness of external files and enables the detection of data tampering. If a hash function with more bits is selected or if a problem occurs with linking to a <uri> element, a data lake that exists for external files can be used to mostly achieve file identification by looking up the corresponding file hash value.

Note 1 The SHA-1 function uses 160-bit (20-byte) hash values, but it has already been shown to crash so preferably should not be used. Use a SHA-2 function such as SHA-256 or SHA-512.

(4) If the external file is in the MaiML data format, indicate the <uuid> element.

Obtain the <uuid> element value from the content in the <document> element included in the MaiML data format external file and use it as the <uuid> element value in the <insertion> element.

That will enable the UUID value, which indicates the uniqueness of MaiML files, to match the cited file in a search of the data lake where the MaiML file is stored, even if the link based on the <uri> elements does not match.

(5) Specify the External File Format

Specify the external file format as a <format> element based on the media type indicated with a multipurpose internet mail extension (MIME), as specified in **JIS B.10.4**.

In **JIS Table B.8**, "application/maiml" and "application/maiml.ZIP" formats are specified specifically for MaiML files. That is consistent with the fact that they contain a <uuid> element.

(6) Making External Files Confidential

Using <childUri>, <childHash>, or <childUuid> elements to apply confidentiality to elements with an <insertion> descendant element ensures their link to the file can be traced.

7. Data Acquisition and Management Technologies for Ensuring Confidentiality and Safety

The remarkable advancements in data encryption and electronic signature technologies have made it difficult to decide requirements for **JIS** standards. Therefore, it was considered desirable to keep usage policies organized within the guideline by maintaining an ongoing exchange of opinions based on actual practices. Therefore, it was specified that it must be possible to indicate <EncryptedData> elements for encryption and <Signature> elements for applying electronic signatures.

Consequently, to enable information confidentiality in MaiML files, XML confidentiality technology can be used to replace information with <EncryptedData> elements in order to enable encryption. However, as discussed for the <insertion> element, if <insertion> elements are included for the purpose of linking to external file and detecting data tampering, then <uri>, <hash>, and <uuid> elements must be written as <childUri>, <childHash>, and <childUuid> elements, respectively.

It was also specified that <Signature> elements must be included as electronic signature technology and "enveloped signature" must be indicated in child elements for <document> elements. Indicating "enveloped signature" provides a way to indicate that an electronic signature generated from the overall file, which does not include the <Signature> element, is written into the XML file.

In terms of electronic signature and encryption capabilities in MaiML files, hash values can be generated with a hash function at least as recent as the SHA-2 function used in XML1.1. Overall, MaiML conforms to XML1.0, but uses electronic signature and encryption functionality specified for XML1.1. The reason is due to clear awareness of crashing problems in the SHA-1 function used for XML1.0.

7.1 Safety of Overall Data

To prevent data tampering or substitution, XML signatures (enveloped signatures) can be applied to overall data files and <Signature> elements can be included in the content of <document> elements.

An example is indicated below.

Example 1

```
<document id="MaimlSampleDoc">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
      <SignatureMethod Algorithm="http://www.w3.org/2009/xmldsig11#dsa-sha256"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <DigestValue>ADY39A1NivZGo9B219Q4jJsDzFE8mktCQze+TTRcOzk=</DigestValue>
    </Reference>
   </SignedInfo>
   <SignatureValue>
    AuVWPnyus62ftJke1L8F9DNn+MwbAKEkBtgKMAOLy1L8+WtlObdnzA==
   </SignatureValue>
  </Signature>
  <uuid>a2be2526-dc71-4035-aeda-e56efb823cb4</uuid>
remainder omitted
```

Of the <Signature> elements, hash values for <DigestValue> elements are generated by the hash function specified by the <DigestMethod> element, but an API is provided for electronic signatures, which is more convenient to use for actual implementation in languages such as JAVA or C#.

The following is an example of using the enveloped signature in JAVA. It is used to create a <Signature> element described above.

```
import javax.xml.crypto.*;
import javax.xml.crypto.dsig.*;
import javax.xml.crypto.dom.*;
import javax.xml.crypto.dsig.dom.DOMSignContext;
import javax.xml.crypto.dsig.keyinfo.*;
import javax.xml.crypto.dsig.spec.*;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.security.*;
import java.util.Collections;
import java.util.Iterator;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;

public class signature {
    public static void main(String[] args) throws Exception {

        // This generates a DOM XMLSignatureFactory for creating enveloped signature data.
        XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");

        // This generates an instance for <Reference>.
        Reference ref = fac.newReference(
                "",
                fac.newDigestMethod(DigestMethod.SHA256, null),
                Collections.singletonList(
                        fac.newTransform(
                                Transform.ENVELOPED,
                                (TransformParameterSpec)null)),
                null,
                null);

        // This generates an instance for <SignedInfo>.
        SignedInfo si = fac.newSignedInfo(
                fac.newCanonicalizationMethod(
                        CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,
                        (C14NMethodParameterSpec)null),
                fac.newSignatureMethod(SignatureMethod.DSA_SHA256,null),
                Collections.singletonList(ref));
        // This generates a DSA key pair.
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
        kpg.initialize(512);
        KeyPair kp = kpg.generateKeyPair();

        // This generates an instance for <KeyValue>.
        // <KeyValue> stores a public key from a generated key pair.
        KeyInfoFactory kif = fac.getKeyInfoFactory();
        KeyValue kv = kif.newKeyValue(kp.getPublic());

        // This generates an instance for <KeyInfo> and stores a key.
        KeyInfo ki = kif.newKeyInfo(Collections.singletonList(kv));

        // This generates an instance for storing XML to be signed.
        // This instance is generated from a signed document.
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setNamespaceAware(true);
        Document doc = dbf.newDocumentBuilder().parse(new FileInputStream("test.xml"));

        // This generated DOMSignContext.
        // This specifies a private key and parent element.
        DOMSignContext dsc = new DOMSignContext(kp.getPrivate(), doc.getDocumentElement());

        // This generates an XML signature instance and specifies signature information and key
        information.
        XMLSignature signature = fac.newXMLSignature(si, ki);

        // Signature
        signature.sign(dsc);
        // This stores the signed XML document in a file.
        OutputStream os = new FileOutputStream("Enveloped.xml");
        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer trans = tf.newTransformer();
        trans.transform(new DOMSource(doc), new StreamResult(os));
    }
}
```

Note 1 Check the applicable programming language manual to confirm that XML signature methods using the SHA-256 algorithm.

Note 2 JIS only specifies the ability to use XML signature technology for detecting data tampering. Signatures based on using asymmetric keys only guarantee that MaiML files were not altered after they were signed. Refer to the FAQ section for information about how to prevent tampering methods such as attackers deleting the XML signature in a MaiML data file and then signing the file again after tampering with the data or attackers falsifying the signer name (typically the owner organization), changing the creation date, or signature date.

Note 3 For signature names, MaiML data file hash values are encrypted with a secret asymmetric key and are verified by decrypting the hash value with a public key and comparing it to a recalculated hash value. That is why a public key is attached to XML signatures. However, never attach a secret key because an attacker could use that same key to apply a signature.

7.2 Ensuring Confidentiality

In order to ensure the independent availability and reproducibility of measurement/analysis data, MaiML files should preferably contain all the information necessary for performing measurements/analyses using the same conditions. However, vendors or users may not want to disclose certain information in some cases.

Information for ensuring the reproducibility of measurements/analyses is indicated in the <protocol> element. In particular, measurement/analysis conditions are indicated in <conditionTemplate> elements with the expectation that they will be used to perform the measurement/analysis again using the same conditions. However, some of those conditions can be kept confidential, such as instrument-specific conditions that are confidential for the vendor or instrument adjustment conditions that the user does not want to disclose.

To prevent disclosing a certain portion of measurement/analysis data, such as intermediate data, that data can be kept separated in <resultTemplate> or <result> elements or encrypting only that data.

In this case, XML element encryption technology is used to encrypt only specific portions of data. Preferably, a fully decodable encryption method should be used.

The following is an example of confidentiality applied using C#.

```
using System;
using System.Xml;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;

namespace CSCrypto {
  class Program {
    static void Main(string[] args)
    {
      Aes key = null;

      try  {
        // Create a new AES key.
        key = Aes.Create();
        // Load an XML document.
        XmlDocument xmlDoc = new XmlDocument() {
          PreserveWhitespace = true
        };
        xmlDoc.Load("test.xml");

        // Encrypt the "peoperty" element.
        Encrypt(xmlDoc, "property", key);
        Console.WriteLine(xmlDoc.InnerXml);

        Decrypt(xmlDoc, key);
        Console.WriteLine(xmlDoc.InnerXml);
      }
      catch (Exception e) {
        Console.WriteLine(e.Message);
      }
      finally  {
        // Clear the key.
        if (key != null) {
          key.Clear();
        }
      }
    }

    public static void Encrypt(XmlDocument Doc, string ElementName, SymmetricAlgorithm Key)
    {
      // Check the arguments.
```

```
        //ArgumentNullException.ThrowIfNull(Doc);
        //ArgumentNullException.ThrowIfNull(ElementName);
        //ArgumentNullException.ThrowIfNull(Key);

        ////////////////////////////////////////////////
        // Find the specified element in the XmlDocument object and create a new XmlElement object.
        ////////////////////////////////////////////////
        Console.WriteLine("Encrypt Method. ElementName = " + ElementName);

        XmlElement elementToEncrypt = Doc.GetElementsByTagName(ElementName)[0] as
        XmlElement;
        // Throw an XmlException if the element was not found.
        if (elementToEncrypt == null) {
            throw new XmlException("The specified element was not found");
        }

        //////////////////////////////////////////////////
        // Create a new instance of the EncryptedXml class and use it to encrypt the XmlElement with
        the
        // symmetric key.
        //////////////////////////////////////////////////

        EncryptedXml eXml = new EncryptedXml();

        byte[] encryptedElement = eXml.EncryptData(elementToEncrypt, Key, false);
        ////////////////////////////////////////////////
        // Construct an EncryptedData object and populate
        // it with the desired encryption information.
        ////////////////////////////////////////////////

        EncryptedData edElement = new EncryptedData(){
            Type = EncryptedXml.XmlEncElementUrl
        };

        string encryptionMethod = null;

        if (Key is Aes) {
            encryptionMethod = EncryptedXml.XmlEncAES256Url;
        } else {
            // Throw an exception if the transform is not AES
            throw new CryptographicException("The specified algorithm is not supported or not
recommended for XML Encryption.");
        }

        edElement.EncryptionMethod = new EncryptionMethod(encryptionMethod);

        // Add the encrypted element data to the
        // EncryptedData object.
        edElement.CipherData.CipherValue = encryptedElement;

        ///////////////////////////////////////////////////
        // Replace the element from the original XmlDocument
        // object with the EncryptedData element.
```

```
            //////////////////////////////////////////////////
            EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
        }

        public static void Decrypt(XmlDocument Doc, SymmetricAlgorithm Alg)
        {
            // Check the arguments.
            //ArgumentNullException.ThrowIfNull(Doc);
            //ArgumentNullException.ThrowIfNull(Alg);

            // Find the EncryptedData element in the XmlDocument.
            XmlElement encryptedElement
             = Doc.GetElementsByTagName("EncryptedData")[0] as XmlElement;

          // If the EncryptedData element was not found, throw an exception.
            if (encryptedElement == null) {
                throw new XmlException("The EncryptedData element was not found.");
            }
          // Create an EncryptedData object and populate it.
            EncryptedData edElement = new EncryptedData();
            edElement.LoadXml(encryptedElement);
          // Create a new EncryptedXml object.
            EncryptedXml exml = new EncryptedXml();
          // Decrypt the element using the symmetric key.
            byte[] rgbOutput = exml.DecryptData(edElement, Alg);
          // Replace the encryptedData element with the plaintext XML element.
            exml.ReplaceData(encryptedElement, rgbOutput);
        }
    }
}
```

Encrypted XML file

```
<xml>
    <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
        <CipherData>
            <CipherValue>
tuAaaiop+65C6YaYJrqh0vMLdVR+yX0HPkwJ/eG+NXgDmYh61brGyXlt5RpyUxPbpH8prZQvverhpIeP
L3q6TPqMvbeMBnFbZ6syi6pCk/UwnWzteJP7S3Y71TEtNvMsInYz5rBxPkPB8ZjYqDKTnDk4ia4v8sQ
Z/RX9MRVyDr31dTwswtIrcNMmWp5R0bnlKFifcHkAwXUdlR+3Ie/gfS2EPRCwMLiNQOcCfEX1Aixi8L
m/6jIk0DYiOad/L7uX
            </CipherValue>
        </CipherData>
    </EncryptedData>
</xml>
```

Example of decrypting an encrypted file

```
<xml>
    <property key="exm:MaterialLotNo" type="stringType">
        <value>Lot No.</value>
        <property key="exm:LotNumber" type="stringType" />
    </property>
</xml>
```

Note 1 Check the applicable programming language manual to confirm XML encryption methods using the SHA-256 algorithm.

Note 2 **JIS** only specifies the ability to use XML encryption technology for keeping data confidential. The examples above use a symmetric common key for applying and removing confidentiality to/from data within the same organization, while also strictly preventing the common key from being leaked outside the organization. Refer to the FAQ section for information about confidentiality policies for disclosing data outside the organization.

7.3 Confidentiality Use Cases

Caution is required for applying confidentiality, because it must be applied in accordance with UPA (Unique Particle Attribution) constraints. (Refer to **JIS K 0200 B.9.2**.)

Confidentiality can be applied to either global elements or content in general purpose data containers. If confidentiality is applied to a global element, it must not be applied to the <uuid> element and it must be applied to the entire remaining portion of the element at one time. To apply confidentiality to the content of a general purpose data container, confidentiality must be applied to all the content at one time. Consequently, to apply confidentiality to only part of a global element, the global element must be separated into elements that may be disclosed and elements that must be kept confidential. (Refer to **JIS K 0200 B.9.2, Example 3**.)

7.4 Preventing Information Leakage or Tampering

MaiML includes tags for electronic signatures (<Signature> element), confidentiality (<Encrypted> element), and for preventing information tampering (<chain>, <hash>, and <uuid> elements). They provide at least the information leakage prevention level provided by the systems suggested by others, such as Adobe for PDF files and by Microsoft for the docx file format.

In particular, the system for using <chain> elements to detect tampering based on the links between files was provided in response to recent cases of fraud and quality falsification based on data tampering in the measurement/analysis industry. Consequently, by implementing the system as specified, data tampering can be detected as a data quality assurance measure.

In contrast, because MaiML was intended to provide independent availability, the disclosure of measurement/analysis information is also expected. That means care must be taken regarding confidential or other information in a company or other organization being leaked outside the organization for some sort of reason. Therefore, data in MaiML files can be encrypted using <Encrypted> elements. The MaiML format does not specify such encryption methods, but rather it specifies that techniques typically used by XML-based formats are used. For information about the corresponding possibility of information leakage and how to prevent such information leakage, refer to the FAQ information indicated in section **11.4**.

Because improvements to encryption using the <Encrypted> element are constantly being proposed, to ensure MaiML does not become an impediment to such improvements, the JIS regulation only specifies that the <Encrypted> element be used and only indicates corresponding descendant elements in examples.

Some of the main causes of information leakage are:

1. Cyber-attacks by unauthorized access, infection by malware, security vulnerabilities, etc.
2. Human error, such as sending email to the wrong recipient, losing a device, etc.
3. Internal fraud intended for wrongful use of information

Such causes indicate a problem with how a database or other system is utilized or operated, rather than a problem with the format itself. They also indicate factors to consider when generating or storing data.

Furthermore, if creating information for disclosure, even if only expected to be used internally within a company or other organization, it is permitted to use elements with a private namespace defined (key attributes in a general purpose data container or <name> elements in a global element) and then delete the tags before disclosure. It also indicates that there are methods available for addressing information leakage by respective companies or organizations providing appropriate filters for disclosure. However, MaiML does not specify how such appropriate filters should be created.

In such cases as well, the relationship to the MaiML file before deletion can be indicated using a <chain> element and including UUID and hash values from the original MaiML file that can be used to detect data tampering.

Therefore, in order to prevent information leakage when using MaiML, users are expected to take responsibility for using the information leakage prevention measures offered by the format to achieve the necessary information leakage prevention levels.

8. Thesaurus and Ontology

Preferably, measurement/analysis terminology used in MaiML should be officially decided and indicated as an ontology in advance. However, due to the diversity in applicable content, it can be difficult to specify the format after deciding the terminology, so that terminology collection never gets started. Therefore, the intention is to use namespaces, <name> elements, and key attributes to ensure the uniqueness of terminology and then keep aggregating differences between companies by gathering names or other means.

Recommended keywords and words used in common for respective instruments will continue to be collected. After that, the terms will be expressed as a graphical ontology and organized to create a thesaurus.

8.1 Terminology Usage Methods

8.1.1 Using Defined Terminology

The MaiML regulation was defined based on the presumption that it would be difficult to specify a unified standard for a diverse variety of measurement/analysis instruments and programs. On the other hand, as more MaiML files are obtained from cyberspace or other sources, it is anticipated that collections of terminology used for the diversity of measurement/analysis instruments and programs will be compiled. When that occurs, it is expected that rhetoric content in namespaces will enable a clear separation of terminology, so that the collection of terminology can be processed based on an ontology of the similar terms obtained and integrated.

On the other hand, from the perspective of users, it would be preferable to have a unified collection of terminology used (key attributes in a general purpose data container or <name> elements in a global element). Therefore, the following describes how defined terminology is used.

8.1.1.1 How to Use Terminology Specified in JIS/ISO or Other Sources

How to define namespaces using URL values or regulation numbers when using terminology specified in an ISO/JIS standard or other source. If defined in a document with a defined URN value, such as “doi,” then the URN value should be used.

Example 1 Using a **JIS** Namespace

<maiml xmlns:JISK0212="https://www.jisc.go.jp/JISK0212:2016/">

After defining the namespace as indicated above, writing “JIS K 0212:absorbance” will enable the namespace to be treated as an expression of absorbance specified by **JIS K 0212** (Technical terms for analytical chemistry (optical part)).

Example 2 Using an ISO namespace

<maiml xmlns:ISO3534="https://www.iso.org/obp/ui/#iso:std:iso:3534:-2:ed-2:v1:en">

After defining the namespace as indicated above, writing "**ISO3534**:accuracy" will enable the namespace to be treated as an expression of accuracy specified in **ISO 3534-2**:2006 (Statistics-Vocabulary and symbols - Part 2:Applied statistics).

8.1.1.2 Thesaurus Available in MaiML

There is a possibility that the organization that administers MaiML or another organization will specify common terminology in the future. If that occurs, it will be possible to post common terminology definitions on the Internet so that the corresponding file URL can be used to specify namespaces and use them as a closed space. However, in that case it will be necessary to carefully identify definition files by managing file versions.

If terminology is specified in a **JIS** or **ISO** standard, then it can be cited by the methods indicated in **8.1.1.1**.

8.1.1.3 Defining Terminology by a MaiML File Creator

It is expected that namespaces will be defined and used by the person, or their affiliated organization, that created the namespace using URN values that enable that person/organization to be identified. Personal URN values include an ORCHID value, which is also used in **8.2**. If the corresponding person is permitted to disclose the ORCHID value, then that can also be used.

8.2 Defining Terminology by a User

If a user wants to define a term, a URN value that can identify the user shall be used. In particular, the following examples describe specifying a disclosed URN, such as an ORCHID value that can identify a researcher, as a namespace.

Example 1 Using a Namespace Defined by a Person Using an ORCHID Value

```
<maiml xmlns:MyWord="https://orcid.org/0000-0002-2494-9603">
…
<property key key="MyWord:SpecialTemperature"> … </property>
```

In this case, MyWord is defined as the namespace. Eventually, the term in the namespace is deployed based on a defined URN value, so it can be defined as a separate keyword.

Example 2 Using a Namespace Defined by an Organization that Uses a DNS Value

```
<maiml xmlns:MyCompany="https://www.aUserCoop.jp/">
…
<property key key="MyCompany:SpecialTemperature"> … </property>
```

In this case, MyCompany is defined as the namespace. Eventually, the term in the namespace is deployed based on a defined URN value, so it can be defined as a separate keyword.

8.3 How to Indicate Relationships between Terms—Expressing Ontologies

Preferably, keywords newly generated by respective vendors should be expressed by a corresponding namespace or a namespace specified separately.

9. Example of Process Flow for Creating Converters for Measurement/Analysis MaiML Files

This chapter uses a single measurement/analysis case as a reference for indicating factors that should be considered, according to the following procedure, when creating a converter for using data output from a specific measurement/analysis by vendor A to create files that include protocol, data, metadata, or other information (collectively referred to as "data" below).

1. Basic naming policies related to data frameworks (section **9.1**)
2. Overview of modeling protocols for measurement/analysis methods (section **9.2**)
3. Naming global elements in level 1 or lower levels (section **9.3**)
4. Naming based on the relationship between <protocol> and <eventLog> elements
   (a) Naming <method> or <log> elements in level 2 (section **9.4**)
   (b) Naming <program> or <trace> elements in level 3 (section **9.5**)
5. Naming based on the relationship between <protocol>, <data>, and <eventLog> elements
   (a) Naming <instruction>, <results>, <event>, or <resultsRef> elements (section **9.6**)
   (b) Naming template or <templateRef> elements (section **9.7**)
   (c) Naming instance or <instanceRef> elements (section **9.8**)
6. Naming related to the process flow of <protocol> elements
   (a) Naming <pnml>, <transition>, and <transitionRef> elements
   (b) Naming <place> and <placeRef> elements
   (c) Naming <arc> elements

9.1 Deciding Basic Policies for Naming

MaiML files are configured with named global elements, general purpose data containers, data process flows, and so on. More specifically, those elements include a <name> element and id and key attributes as data types (names). That means vendors must be careful when assigning names. Decide names based on the XML identifier naming guideline in Appendix B.

Table 9.1 Elements and Attributes that should be Considered for Naming

Named Element/Attribute	Purpose of Use
<name> element	Indicates the meaning of global element.
id attribute	Indicates the relationship between global elements in files.
key attribute	Indicates the meaning of individual data contained in a general purpose data container.

This is an example of vendor A assigning XML modifier names (<name> elements and id attributes) corresponding to data structure names, class names, and database table names (indicated as "structure names" below) to global elements and other MaiML elements that include a <name> element, based on the data structures contained in a particular measurement/analysis "XXX."

In this case, the structure used internally in the original data format refers collectively to the sample information, measurement/analysis conditions, measurement/analysis results, and other data as a single set of data, where each structure is assumed to correspond to a <materialTemplate>, <conditionTemplate>, <resultTemplate>, or other global element. Each structure consists of a set of metadata that includes the individual parameter name paired with corresponding data.

Note 1 For example, tiff format image files can be thought of as a MaiML file global element with a structure consisting of a set of IFD (Image File Directory) metadata. Inside that structure are multiple tags (directory entries) for the data name, image data type, and data value, which form a data set. That data corresponds to the key attribute, xsi:type attribute, and <value> element contained in a general purpose data container.

First, the <name> element and key attribute are expected to be used as terms in a thesaurus utilized by RDF/OWL vocabularies. Consequently, it is difficult to immediately determine the appropriateness of the structure names, class names, and database table names in the data structures used thus far for the original measurement/analysis "XXX." Therefore, the structure, class, and database table names used in the source data structure are first assigned English descriptions in <description> elements. Then the <name> elements and id attributes are named based on those descriptions.

In this case, the "XXX" characters at the beginning of the English descriptions in <name> elements are replaced with a prefix in the form xxx, other portions are named according to PascalCase naming rules and the id attributes are named with the element name at the beginning and according to the snake_case naming rules, but other naming rules may be used instead.

Note 2 In this case, "xxx" indicates a prefix assigned to <name> elements of global elements relevant to the "XXX" measurement/analysis method specified by the vendor organization, in accordance with the xmlns:xxx="http://www.examplevendor.com/ontology/maiml/XXX#" statement.

9.2 Overview of Modeling Protocols for Measurement/Analysis

MaiML files require organizing applicable measurement/analysis methods as a flowchart using a Petri net (section **3.3**). When vendors and MaiML file users review the flowchart after section **9.3**, the flowchart will also result in designing more appropriate measurement/analysis process flows.

Fig. 9.1 shows an example of a protocol for the "XXX" measurement/analysis specified by the organization at vendor A.

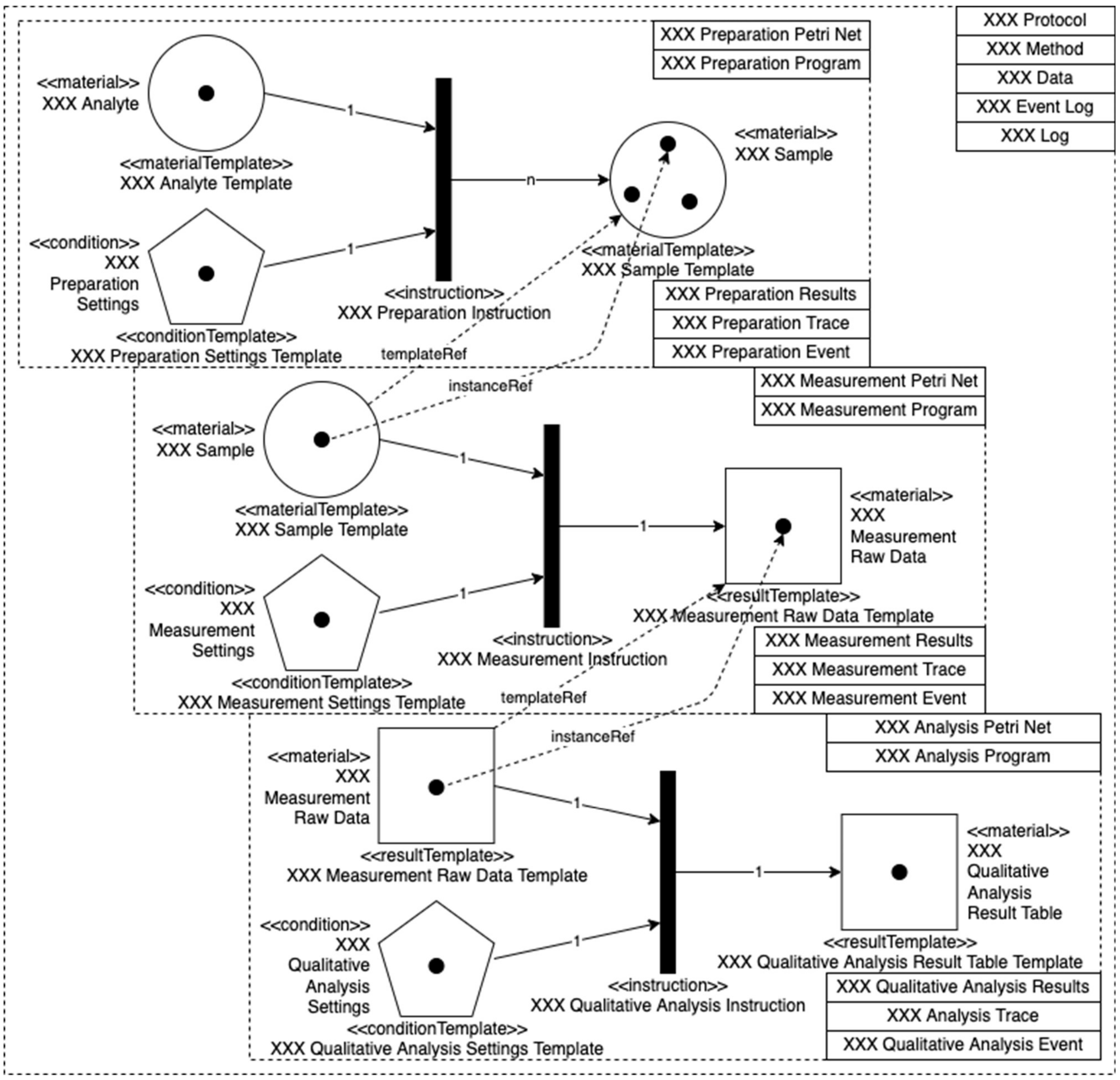


Fig. 9.1 Example of Protocol and Global Element Naming for Measurement/Analysis Method XXX

Fig. 9.1 is a Petri net, described in **3.3**, of the data and metadata framework determined in section **9.1** and organized into sample information (section **4.3.2.5**), measurement/analysis conditions (section **4.3.2.6**), and measurement/analysis results (section **4.3.2.7**) classifications. By focusing on the measurement/analysis operations, the information in the <instruction> element (section **4.3.2.4**) was classified as either input information or output information, as described in section **9.4**. If part of the information is expected to be kept confidential, it can be divided into data and metadata frameworks (corresponding to <place> elements) based on the stage when confidentiality is applied. In **Fig. 9.1**, confidentiality is either applied to the entire process flow or the process flow is expressed as one framework to enable either disclosing or not disclosing the information.

Furthermore, assuming n = 3 tokens are generated by the output from the first process, **Fig. 9.1** assigns three tokens, but for subsequent processes, it shows the case of only the first token being processed. Because it also shows consumed tokens, it is not actually a proper Petri net, but it does show the correspondence between the sample information instance, condition information instance, and results information instance, so all tokens are indicated.

9.3 Naming Global Elements in Level 1 or Lower Levels

The global elements in the first hierarchical level in MaiML files used to store XXX measurement/analysis data could be named as shown below, for example.

Table 9.2 Naming Global Elements

<document> element		
XPath	/maiml/document	
<description> element	XXX Data File Document	Description of MaiML file document, including XXX measurement/analysis method results and event log
<name> element	xxx:DataFileDocument	Name (XML modification name) of MaiML file document, including XXX measurement/analysis method results and event log
id attribute	document_xxx_data_file	ID in XML file for MaiML file document, including XXX measurement/analysis method results and event log
<protocol> element		
XPath	/maiml/protocol	
<description> element	XXX Protocol	Description of protocol for XXX measurement/analysis method
<name> element	xxx:Protocol	Name (XML modification name) of protocol for XXX measurement/analysis method
id attribute	protocol_xxx	ID in XML file for protocol for XXX measurement/analysis method
<data> element		
XPath	/maiml/data	
<description> element	XXX Data	Description of data for XXX measurement/analysis method
<name> element	xxx:Data	Name (XML modification name) of data for XXX measurement/analysis method
id attribute	data_xxx	ID in XML file of data for XXX measurement/analysis method
<eventLog> element		
XPath	/maiml/eventLog	
<description> element	XXX Event Log	Description of eventLog for XXX measurement/analysis method
<name> element	xxx:EventLog	Name (XML modification name) of eventLog for XXX measurement/analysis method
id attribute	eventLog_xxx	ID in XML file of eventLog for XXX measurement/analysis method

Note 1 XPath is a method of indicating XML hierarchical structures in terms of a directory tree, with routing elements used to show absolute paths or axis elements used to show relative paths. It is used to indicate XML structures without XML code or to search for data. Attributes are indicated with an @ symbol at the beginning of attribute names. If identical element names are listed, a conditional expression is added to the end of element names to qualify them. For example, XPath expressions “protocol/method[1]” and “protocol/method[position()=1],” which show the <maiml> element as an axis, indicate the first <method> element, which is a child element of a <protocol> element, whereas the “protocol/method[@id='method_xxx']” expression indicates the <method> element with a “method_xxx” value for the id attribute, which is a child element of the <protocol> element.

Note 2 MaiML files only contain one of each element in level 1, so except for the <document> element, <description> elements have the form “XXX element name,” <name> elements are converted directly to camelCase, and id attributes are converted to snake_case with the element name at the beginning. Because there are two notation methods for indicating the <description> element, <name>, and id attribute for <document> elements in MaiML files, either by indicating only the <document> and <protocol> elements or by also indicating

<data> and <eventLog> elements, notation was indicated in a way that enables distinguishing between the two methods.

9.4 Naming <method> or <log> Elements in Level 2

After the measurement/analysis method is determined, the next step is to decide the protocol (process). That can be decided by determining the order respective data stored in structure names, class names, or database table names in the original data structure that correspond to the <instruction> elements for measurement/analysis operations were created.

Measurement/analysis process flows include repeatedly executable processes that are indicated in <protocol> elements (section **4.3.2**).

Measurement/analysis process flows can include one or more (normally one) collections of processes that are indicated in <method> elements that are child elements of a <protocol> element (section **4.3.2.1**). In this case, the XXX measurement/analysis includes one <method> element.

A collection of processes corresponding to one log is indicated in a <log> element that is a child element of an <eventLog> element and the corresponding ref attribute references the id attribute for a <method> element.

At this point, the <method> and <log> elements in the MaiML file where the XXX measurement/analysis data is stored can be named as indicated below, for example.

Table 9.3 Examples of Naming <method> and <log> Elements

<method> element		
XPath	/maiml/protocol/method[1]	
<description> element	XXX Method	Description of method for XXX measurement/analysis method
<name> element	xxx:Method	Name of method for XXX measurement/analysis method (XML modified name)
id attribute	method_xxx	ID in XML file of method for XXX measurement/analysis method
<log> element		
XPath	/maiml/eventLog/log[1]	
<description> element	XXX Log	Description of log for XXX measurement/analysis method
<name> element	xxx:Log	Name of log for XXX measurement/analysis method (XML modified name)
id attribute	log_xxx	ID in XML file of log for XXX measurement/analysis method
ref attribute	method_xxx	References the ID in XML file of method for XXX measurement/analysis method

9.5 Naming Level 3 <program> or <trace> Elements

A collection of processes can contain one or more series of processes and are indicated in a <program> element that is a child element of a <method> element. Process series candidates include processes such as sample preparation (Preparation), measurement (Measurement), and analysis (Analysis). In this example, the process series includes three <program> elements, which are sample preparation (Preparation), measurement (Measurement), and analysis (Analysis), as indicated in **Fig. 9.1**.

Each series of processes corresponds to a log file for one series of processes and is indicated in a <trace> element that is a child element of a <log> element with a corresponding ref attribute that references the id attribute of the <program> element.

At this point, the <program> and <trace> elements for the MaiML file where the XXX measurement/analysis method data is stored can be named as indicated below, for example.

Table 9.4 Examples of Naming <program> and <trace> Elements

<program> element		
XPath	/maiml/protocol/method[1]/program[1]	
<description> element	XXX Preparation Program	Description of program for sample preparation
<name> element	xxx:PreparationProgram	Name of program for sample preparation (XML modified name)
id attribute	program_xxxPreparation	ID in XML file of the sample preparation program
<program> element		
XPath	/maiml/protocol/method[1]/program[2]	
<description> element	XXX Measurement Program	Description of program for measurement
<name> element	xxx:MeasurementProgram	Name of program for measurement (XML modified name)
id attribute	program_xxxMeasurement	ID in XML file of the measurement program
<program> element		
XPath	/maiml/protocol/method[1]/program[3]	
<description> element	XXX Analysis Program	Description of program for analysis
<name> element	xxx:AnalysisProgram	Name of program for analysis (XML modified name)
id attribute	program_xxxAnalysis	ID in XML file of the analysis program
<trace> element		
XPath	/maiml/eventLog/log[1]/trace[1]	
<description> element	XXX Preparation Trace	Description of trace for sample preparation
<name> element	xxx:PreparationTrace	Name of trace for sample preparation (XML modified name)
id attribute	trace_xxxPreparation	ID in XML file of the sample preparation trace
ref attribute	program_xxxPreparation	References the ID in the XML file of program for sample preparation
<trace> element		
XPath	/maiml/eventLog/log[1]/trace[2]	
<description> element	XXX Measurement Trace	Description of trace for measurement
<name> element	xxx:MeasurementTrace	Name of trace for measurement (XML modified name)
id attribute	trace_xxxMeasurement	ID in XML file of the measurement trace
ref attribute	program_xxxMeasurement	References the ID in the XML file of program for measurement
<trace> element		
XPath	/maiml/eventLog/log[1]/trace[3]	
<description> element	XXX Analysis Trace	Description of trace for analysis
<name> element	xxx:AnalysisTrace	Name of trace for analysis (XML modified name)
id attribute	trace_xxxAnalysis	ID in XML file of the analysis trace
ref attribute	program_xxxAnalysis	References the ID in the XML file of program for analysis

9.6 Examples of Naming <instruction>, <results>, <event>, and <resultsRef> Elements

A process series can include one or more operations, which are indicated in <instruction> elements that are child elements of a <program> element.

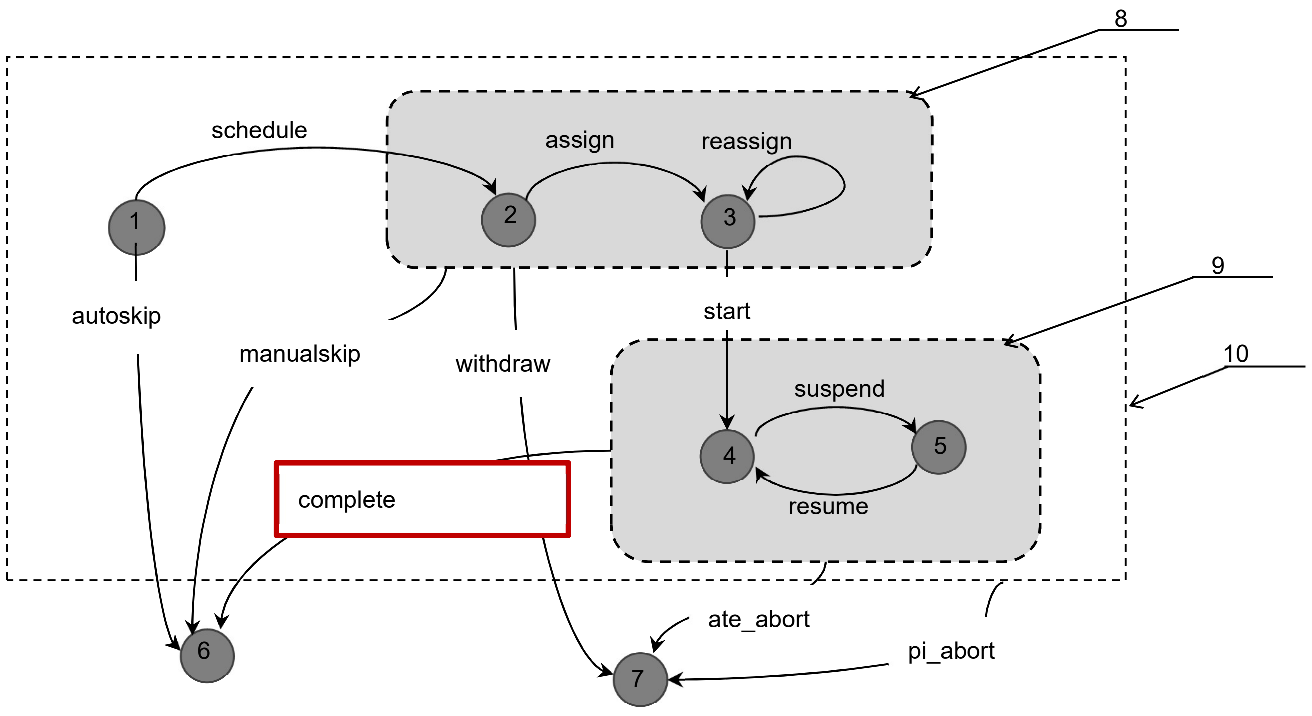
In this case, if there is one operation in a series of processes, then operations such as sample preparation (Preparation), measurement (Measurement), and analysis (Analysis) may be used as candidate operations. To indicate a process series with at least one operation in more detail, the processes may be indicated separately for weighing, dissolution, dilution, qualitative analysis, quantitative analysis, image analysis, or other processes. In this example, the process series includes <instruction> elements for sample preparation (Preparation), measurement (Preparation), and qualitative analysis (Qualitative Analysis) processes.

A combination of sample, condition, and results information instances on input and output sides must be clearly indicated for each operation and indicated collectively in the <results> element that is a child element of the <data> element.

Accordingly, each operation must be indicated in an <event> element that is a child element of the <trace> element for one or more operation logs. The id attribute of the <instruction> element is referenced based on the ref attribute of indicated <event> elements.

Furthermore, the relationships between operation logs and the combination of sample, condition, and results information instances on input and output sides of operations can be clearly indicated in <resultsRef> elements that are child elements of an <event> element, with the corresponding ref attribute referencing the id attribute of the <results> element.

Note 1 Because operations are repeatedly executable, they are treated as a combination of sample, condition, and results information instances on input and output sides of operations. MaiML files might contain more than one log for instances and operations. In this example, the serial number "_i01" is appended to id attributes for <results>, <event>, and <resultsRef> elements, where "i" indicates "index."



Definition of Symbols

1: Initial state	6: Finished state
2: Preparation state	7: Abnormal termination state
3: Ready state	8: Preparation process
4: Action state	9: Action process
5: Sleep state	10: Process being processed

The diagram above shows a model of transitions between measurement/analysis operation states if logs are indicated for measurement/analysis operations. The model conforms to the XES Standard format. The terms shown in the figure ("schedule," "assign," "reassign," "start," "manualskip," "autoskip," "withdraw," "suspend," "resume," "ate_abort"," "pi_abort," and "complete") indicate the type of measurement/analysis operation state transition (refer to **JIS K 0200**, **B.8**). For example, the measurement/analysis operation "complete" indicates that the measurement/analysis process was completed normally.

The circled numbers indicate the measurement/analysis process states and the arrows indicate the transitions from those states. Areas enclosed by a dashed line indicate metastates with multiple states.
The <property> element, which is one type of content included in <event> elements, includes a key attribute “lifecycle:transition” with a <value> element where status transitions like those shown here are indicated.

Fig. 9.2 Measurement/Analysis Operation Status Transition Model (Cited from JIS K 0200, Fig. 7)

Note 2 Operation logs must include the “complete” type of state transition shown in **Fig. 9.2**. Since measurement/analysis operations cannot be started if sample, condition, or results information cannot be prepared as operation inputs, the “schedule,” “assign,” or “start” state transition types shown in **Fig. 9.2** are also sometimes used. In this example, “_complete” was added in front of the serial number appended to id attributes for <event> and <resultsRef> elements and “complete” was indicated in the <value> element for the key attribute “lifecycle:transition” in the <property> element in the general purpose data container in the <event> element. If simultaneously output instance elements are anticipated when complete and a <resultsRef> element is specified, then indicating a <property> element for the key attribute “concept:instance” is not essential.

Note 3 An example of using the “schedule,” “start,” and “complete” types of state transitions shown in **Fig. 9.2** is included in section **9.12**.

At that point, the <instruction>, <results>, <event>, and <resultsRef> elements used in MaiML files for storing XXX measurement/analysis data can be named as follows, for example.

Table 9.5 Examples of Naming <instruction>, <results>, <event>, and <resultsRef> Elements

<instruction> element		
XPath	/maiml/protocol/method[1]/program[1]/instruction[1]	
<description> element	XXX Preparation Instruction	Description of instruction for sample preparation
<name> element	xxx:PreparationInstruction	Name of instruction for sample preparation (XML modified name)
id attribute	instruction_xxxPreparation	ID in XML file of the sample preparation instruction
<instruction> element		
XPath	/maiml/protocol/method[1]/program[2]/instruction[1]	
<description> element	XXX Measurement Instruction	Description of instruction for measurement
<name> element	xxx:MeasurementInstruction	Name of instruction for measurement (XML modified name)
id attribute	instruction_xxx Measurement	ID in XML file of the measurement instruction
<instruction> element		
XPath	/maiml/protocol/method[1]/program[3]/instruction[1]	
<description> element	XXX Qualitative Analysis Instruction	Description of instruction for qualitative analysis
<name> element	xxx:QualitativeAnalysisInstruction	Name of instruction for qualitative analysis (XML modified name)
id attribute	instruction_xxxQualitativeAnalysis	ID in XML file of the qualitative analysis instruction
<results> element		
XPath	/maiml/data/results[1]	
<description> element	XXX Preparation Results	Description of results for sample preparation
<name> element	xxx:PreparationResults	Name of results for sample preparation (XML modified name)
id attribute	results_xxxPreparation_i01	ID in XML file of sample preparation results

<results> element		
XPath	/maiml/data/results[2]	
<description> element	XXX Measurement Results	Description of results for measurement
<name> element	xxx:MeasurementResults	Name of results for measuremen (XML modified name)
id attribute	results_xxxMeasurement_i01	ID in XML file of measurement results
<results> element		
XPath	/maiml/data/results[3]	
<description> element	XXX Qualitative Analysis Results	Description of results for qualitative analysis
<name> element	xxx:QualitativeAnalysisResults	Name of results for qualitative analysis (XML modified name)
id attribute	results_xxxQualitativeAnalysis_i01	ID in XML file of qualitative analysis results
<event> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[1]	
<description> element	XXX Preparation Event	Description of event for sample preparation
<name> element	xxx:PreparationEvent	Name of event for sample preparation (XML modified name)
id attribute	event_xxxPreparation_complete_i01	ID in XML file of event for sample preparation completion
ref attribute	instruction_xxxPreparation	References the ID in XML file of instruction for sample preparation
<event> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[1]	
<description> element	XXX Measurement Event	Description of event for measurement
<name> element	xxx:MeasurementEvent	Name of event for measurement (XML modified name)
id attribute	event_xxxMeasurement_complete_i01	ID in XML file of event for measurement completion
ref attribute	instruction_xxxMeasurement	References the ID in XML file of instruction for measurement
<event> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]	
<description> element	XXX Qualitative Analysis Event	Description of event for qualitative analysis
<name> element	xxx:QualitativeAnalysisEvent	Name of event for qualitative analysis (XML modified name)
id attribute	event_xxxQualitativeAnalysis_complete_i01	ID in XML file of event for qualitative analysis completion
ref attribute	instruction_xxxQualitativeAnalysis	References the ID in XML file of instruction for qualitative analysis
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[1]/resultsRef[1]	
<description> element	XXX Preparation Results Reference	Description of resultsRef for sample preparation
<name> element	xxx:PreparationResultsReference	Name of resultsRef for sample preparation (XML modified name)
id attribute	resultsRef_xxxPreparation_complete_i01	ID in XML file of resultsRef for sample preparation completion
ref attribute	results_xxxPreparation_i01	References the ID in XML file of results for sample preparation
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[1]/resultsRef[1]	
<description>	XXX Measurement Results Reference	Description of resultsRef for

element		measurement
<name> element	xxx:MeasurementResultsReference	Name of resultsRef for measurement (XML modified name)
id attribute	resultsRef_xxxMeasurement_complete_i01	ID in XML file of resultsRef for measurement completion
ref attribute	results_xxxMeasurement_i01	References the ID in XML file of results for measurement
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]/ resultsRef[1]	
<description> element	XXX Qualitative Analysis Results Reference	Description of resultsRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisResultsReference	Name of resultsRef for qualitative analysis (XML modified name)
id attribute	resultsRef_xxxQualitativeAnalysis_complete_i01	ID in XML file of resultsRef for qualitative analysis completion
ref attribute	results_xxxQualitativeAnalysis_i01	References the ID in XML file of results for qualitative analysis

9.7 Naming Template and <templateRef> Elements

For each measurement/analysis operation, three types of template elements must be used to indicate sample information (<materialTemplate> elements), condition information (<conditionTemplate> elements), and results information (<resultTemplate> elements) on input and output sides. Those template elements for sample, condition, and results information are used as templates for indicating information used in repeatedly executed processes and are indicated in <materialTemplate>, <conditionTemplate>, and <resultTemplate> elements that are descendant elements of the <protocol> element.

In this case, they include Analyte Template and Preparation Settings Template used as inputs for sample preparation (Preparation); Sample Template used as outputs for sample preparation (Preparation) and as inputs for measurement (Measurement); Measurement Settings Template used as inputs for measurement (Measurement); Measurement Raw Data Template used as outputs for measurement (Measurement) and inputs for qualitative analysis (Qualitative Analysis); Qualitative Analysis Settings Template used as inputs for qualitative analysis (Qualitative Analysis); and Qualitative Analysis Result Table Template used as outputs for qualitative analysis (Qualitative Analysis).

Note that the descriptions of other global elements describe MaiML-specific classes, so descriptions of template elements for sample, condition, and results information preferably should use common names. Therefore, the abstract class meanings of the terms “material,” “condition,” and “results” were avoided in descriptions.

Template elements may be indicated in any level of <protocol>, <method>, or <program> elements. However, the fact that levels also indicate accessibility to data must be considered. Such considerations are indicated in the notes below.

Note 1 Normally, if the template element serves as the only input or output operation, then it should be indicated as a child element of a <program> element.

Note 2 If a template element serves as both an output for an operation in the previous process and an input for an operation in the subsequent process and also has the same <method> element as the ancestor element of two <instruction> elements with different <program> elements as parent elements, then the template element should either be indicated as the child element of the closest <method> element with a common ancestor as the template element or the template element and its copy element should be indicated as child elements for two <program> elements connected by a <templateRef> element.

Note 3 If a template element serves as both an output for an operation in the previous process an input for an operation in the subsequent process, but has different <method> elements as the ancestor elements of two <instruction> elements, then the template element should be indicated as the child element of the closest <protocol> element with a common ancestor as the template element or the template element and its copy element should be indicated as child elements for two <method> elements connected by a <templateRef> element.

In this case, a template element and its copy element were indicated as the respective child elements of two <program> elements for the previous and subsequent processes, with the id attribute referenced by the ref attribute of the <templateRef> element that is the child element of the template element copy.

Note 4 Because the template element and copy template element both have input and output sides, this example shows "_output" or "_input" appended to the id attributes for template, copy, and <templateRef> elements.

Note 5 The same <description> and <name> elements were used for the template and copy elements (but with different id attributes). Separate names are assigned in some cases for completely different combinations with a general purpose data container instead of a copy element.

At this point, the template element and <templateRef> element used to store XXX measurement/analysis method data in MaiML files can be named as indicated below, for example.

Table 9.6 Examples of Naming Template and <templateRef> Elements

<materialTemplate> element		
XPath	/maiml/protocol/method[1]/program[1]/materialTemplate[1]	
<description> element	XXX Analyte Template	Description of materialTemplate for analyte
<name> element	xxx:AnalyteTemplate	Name of materialTemplate for analyte (XML modified name)
id attribute	materialTemplate_xxxAnalyte_input	ID in XML file of materialTemplate for analyte used as input
<conditionTemplate> element		
XPath	/maiml/protocol/method[1]/program[1]/conditionTemplate[1]	
<description> element	XXX Preparation Settings Template	Description of conditionTemplate for sample preparation condition settings
<name> element	xxx:PreparationSettingsTemplate	Name of conditionTemplate for sample preparation condition settings (XML modified name)
id attribute	conditionTemplate_xxxPreparationSettings_input	ID in XML file of conditionTemplate for sample preparation condition settings used as input
<materialTemplate> element		
XPath	/maiml/protocol/method[1]/program[1]/materialTemplate[2]	
<description> element	XXX Sample Template	Description of materialTemplate for sample

<name> element	xxx:SampleTemplate	Name of materialTemplate for sample (XML modified name)
id attribute	materialTemplate_xxxSample_output	ID in XML file of materialTemplate for sample used as output
<materialTemplate> element		
XPath	/maiml/protocol/method[1]/program[2]/materialTemplate[1]	
<description> element	XXX Sample Template	Description of materialTemplate for sample
<name> element	xxx:SampleTemplate	Name of materialTemplate for sample (XML modified name)
id attribute	materialTemplate_xxxSample_input	ID in XML file of materialTemplate for sample used as input
<conditionTemplate> element		
XPath	/maiml/protocol/method[1]/program[2]/conditionTemplate[1]	
<description> element	XXX Measurement Settings Template	Description of conditionTemplate for measurement condition settings
<name> element	xxx:MeasurementSettingsTemplate	Name of conditionTemplate for measurement condition settings (XML modified name)
id attribute	conditionTemplate_xxxMeasurementSettings_input	ID in XML file of conditionTemplate for measurements condition settings used as input
<resultTemplate> element		
XPath	/maiml/protocol/method[1]/program[2]/resultTemplate[1]	
<description> element	XXX Measurement Raw Data Template	Description of resultTemplate for measurement raw data
<name> element	xxx:MeasurementRawDataTemplate	Name of resultTemplate for measurement raw data (XML modified name)
id attribute	resultTemplate_xxxMeasurementRawData_output	ID in XML file of resultTemplate for measurement raw data used as output
<resultTemplate> element		
XPath	/maiml/protocol/method[1]/program[3]/resultTemplate[1]	
<description> element	XXX Measurement Raw Data Template	Description of resultTemplate for measurement raw data
<name> element	xxx:MeasurementRawDataTemplate	Name of resultTemplate for measurement raw data (XML modified name)
id attribute	resultTemplate_xxxMeasurementRawData_input	ID in XML file of resultTemplate for measurement raw data

		used as input
<conditionTemplate> element		
XPath	/maiml/protocol/method[1]/program[3]/conditionTemplate[1]	
<description> element	XXX Qualitative Analysis Settings Template	Description of conditionTemplate for qualitative analysis condition settings
<name> element	xxx:QualitativeAnalysisSettingsTemplate	Name of conditionTemplate for qualitative analysis condition settings (XML modified name)
id attribute	conditionTemplate_xxxQualitativeAnalysisSettings_input	ID in XML file of conditionTemplate for qualitative analysis condition settings used as input
<resultTemplate> element		
XPath	/maiml/protocol/method[1]/program[3]/resultTemplate[2]	
<description> element	XXX Qualitative Analysis Result Table Template	Description of resultTemplate for qualitative analysis results table
<name> element	xxx:QualitativeAnalysisResultTableTemplate	Name of resultTemplate for qualitative analysis results table (XML modified name)
id attribute	resultTemplate_xxxQualitativeAnalysisResultTable_output	ID in XML file of resultTemplate for qualitative analysis results table used as output
<templateRef> element		
XPath	/maiml/protocol/method[1]/program[2]/materialTemplate[1]/templateRef[1]	
<description> element	XXX Sample Template Reference	Description of templateRef for sample
<name> element	xxx:SampleTemplateReference	Name of templateRef for sample (XML modified name)
id attribute	templateRef_xxxSample_input	ID in XML file of templateRef for samples used as input
ref attribute	materialTemplate_xxxSample_output	References the ID in XML file of materialTemplate for sample used as output
<templateRef> element		
XPath	/maiml/protocol/method[1]/program[3]/resultTemplate[1]/templateRef[1]	
<description> element	XXX Measurement Raw Data Template Reference	Description of templateRef for measurement raw data
<name> element	xxx:MeasurementRawDataTemplateReference	Name of templateRef for measurement raw data (XML modified name)
id attribute	templateRef_xxxMeasurementRawData_input	ID in XML file of templateRef for measurement raw data used as input
ref attribute	resultTemplate_xxxMeasurementRawData_output	References the ID in

		XML file of resultTemplate for measurement raw data used as output

9.8 Naming Instance and <instanceRef> Elements

Each operation must clearly indicate sample, condition, and results information instance elements on input and output sides of the operation. In this case, instance elements for sample, condition, and results information refer to information that ties measurement/analysis data together as one repeatedly executed process flow. They are indicated in <material>, <condition>, and <result> elements that are child elements of a <results> element that is a child element of a <data> element, the corresponding ref attributes references the id attributes for <materialTemplate>, <conditionTemplate>, and <resultTemplate> elements.

In this case, they include Analyte and Preparation Settings used as inputs for sample preparation (Preparation); Sample used as outputs for sample preparation (Preparation) and as inputs for measurement (Measurement); Measurement Settings used as inputs for measurement (Measurement); Measurement Raw Data used as outputs for measurement (Measurement) and inputs for qualitative analysis (Qualitative Analysis); Qualitative Analysis Settings used as inputs for qualitative analysis (Qualitative Analysis); and Qualitative Analysis Result Table used as outputs for qualitative analysis (Qualitative Analysis).

Note that the descriptions of other global elements describe MaiML-specific classes, so descriptions of instance elements for sample, condition, and results information preferably should use common names. Therefore, the abstract class meanings of the terms “material,” “condition,” and “results” were avoided in descriptions.

If an instance element is an output for an operation in the previous process and also an input for an operation in the subsequent process, that would result in two <results> parent elements. Therefore, in this example, the instance element and its copy were included separately as child elements for the two <results> elements in the previous and subsequent processes, with the ref attribute for the <instanceRef> element that is a child element of the instance element copy references the id attribute of the instance element.

Note 1 Because the instance element and its copy both have input and outputsides, this example shows “_output” or “_input” appended to the id attributes for instance, instance copy, and <instanceRef> elements.

Note 2 Because the number of tokens consumed or generated by each firing of a transition in Petri nets is not necessarily one, this example shows the serial number “_m01” added to the id attributes of incidence, incidence copy, and <instanceRef> elements (where “m” refers to “multiplicity”).

Note 3 Because operations can be executed repeatedly, the number of instance, instance copy, or <instanceRef> elements in MaiML files is not necessarily one. This example shows the serial number “_i01” added to the id attributes of incidence, incidence copy, and <instanceRef> elements (where “i” refers to “index”).

Note 4 The <description> and <name> elements for the instance and instance copy elements are assumed to be the same (but with different id attributes). Separate names are assigned in some cases for completely different combinations with a general purpose data container instead of a copy element.

Note 5 In contrast, the relationship between template elements and corresponding instance elements is basically that one is derived from the other, which can be thought of as meaning an extension from or an overwriting of the other.

At this point, instance elements and <instanceRef> elements used to store XXX measurement/analysis method data in MaiML files can be named as indicated below, for example.

Table 9.7 Examples of Naming Instance and <instanceRef> Elements

<material> element		
XPath	/maiml/data/results[1]/material[1]	
<description> element	XXX Analyte	Description of material for analytes
<name> element	xxx:Analyte	Name of material for analytes (XML modified name)
id attribute	material_xxxAnalyte_input_m01_i01	ID in XML file of material for analytes used as input
ref attribute	materialTemplate_xxxAnalyte_input	References the ID in XML file of materialTemplate for analytes used as input
<condition> element		
XPath	/maiml/data/results[1]/condition[1]	
<description> element	XXX Preparation Settings	Description of condition for sample preparation condition settings
<name> element	xxx:PreparationSettings	Name of condition for sample preparation condition settings (XML modified name)
id attribute	condition_xxxPreparationSettings_input_m01_i01	ID in XML file of condition for sample preparation condition settings used as input
ref attribute	conditionTemplate_xxxPreparationSettings_input	References the ID in XML file of conditionTemplate for sample preparation condition settings used as input
<material> element		
XPath	/maiml/data/results[1]/material[2]	
<description> element	XXX Sample	Description of material for sample
<name> element	xxx:Sample	Name of material for sample (XML modified name)
id attribute	material_xxxSample_output_m01_i01	ID in XML file of material for sample used as output
ref attribute	materialTemplate_xxxSample_output	References the ID in XML file of materialTemplate for sample used as output
<material> element		
XPath	/maiml/data/results[2]/material[1]	
<description> element	XXX Sample	Description of material for sample
<name> element	xxx:Sample	Name of material for sample (XML modified name)
id attribute	material_xxxSample_input_m01_i01	ID in XML file of material for sample used as input
ref attribute	materialTemplate_xxxSample_input	References the ID in XML file of materialTemplate for sample used as input
<condition> element		
XPath	/maiml/data/results[2]/condition[1]	
<description> element	XXX Measurement Settings	Description of condition for measurement condition settings

<name> element	xxx:MeasurementSettings	Name of condition for measurement condition settings (XML modified name)
id attribute	condition_xxxMeasurementSettings_input_m01_i01	ID in XML file of condition for measurement condition setting used as input
ref attribute	conditionTemplate_xxxMeasurementSettings_input	References the ID in XML file of conditionTemplate for measurement condition settings used as input
<result> element		
XPath	/maiml/data/results[2]/result[1]	
<description> element	XXX Measurement Raw Data	Description of result for measurement raw data
<name> element	xxx:MeasurementRawData	Name of result for measurement raw data (XML modified name)
id attribute	result_xxxMeasurementRawData_output_m01_i01	ID in XML file of result for measurement raw data used as output
ref attribute	resultTemplate_xxxMeasurementRawData_output	References the ID in XML file of resultTemplate for measurement raw data used as output
<result> element		
XPath	/maiml/data/results[3]/result[1]	
<description> element	XXX Measurement Raw Data	Description of result for measurement raw data
<name> element	xxx:MeasurementRawData	Name of result for measurement raw data (XML modified name)
id attribute	result_xxxMeasurementRawData_input_m01_i01	ID in XML file of result for measurement raw data used as input
ref attribute	resultTemplate_xxxMeasurementRawData_input	References the ID in XML file of resultTemplate for measurement raw data used as input
<condition> element		
XPath	/maiml/data/results[3]/condition[1]	
<description> element	XXX Qualitative Analysis Settings	Description of condition for qualitative analysis condition settings
<name> element	xxx:QualitativeAnalysisSettings	Name of condition for qualitative analysis condition settings (XML modified name)
id attribute	condition_xxxQualitativeAnalysisSettings_input_m01_i01	ID in XML file of condition for qualitative analysis condition settings used as input
ref attribute	conditionTemplate_xxxQualitativeAnalysisSettings_input	References the ID in XML file of conditionTemplate qualitative analysis condition settings used as input
<result> element		
XPath	/maiml/data/results[3]/result[2]	
<description> element	XXX Qualitative Analysis Result Table	Description of result for qualitative analysis results table
<name> element	xxx:QualitativeAnalysisResultTable	Name of result for qualitative

		analysis results table (XML modified name)
id attribute	result_xxxQualitativeAnalysisResultTable_output_m01_i01	ID in XML file of result for qualitative analysis results table used as output
ref attribute	resultTemplate_xxxQualitativeAnalysisResultTable_output	References the ID in XML file of resultTemplate for qualitative analysis results table used as output
<instanceRef> element		
XPath	/maiml/data/results[2]/material[1]/instanceRef[1]	
<description> element	XXX Sample Instance Reference	Description of instanceRef for sample
<name> element	xxx:SampleInstanceReference	Name of instanceRef for sample (XML modified name)
id attribute	instanceRef_xxxSample_input_m01_i01	ID in XML file of instanceRef for sample used as input
ref attribute	material_xxxSample_output_m01_i01	References the ID in XML file of material for sample used as output
<instanceRef> element		
XPath	/maiml/data/results[3]/result[1]/instanceRef[1]	
<description> element	XXX Measurement Raw Data Instance Reference	Description of instanceRef for measurement raw data
<name> element	xxx:MeasurementRawDataInstanceReference	Name of instanceRef for measurement raw data (XML modified name)
id attribute	instanceRef_xxxMeasurementRawData_input_m01_i01	ID in XML file of instanceRef for measurement raw data used as input
ref attribute	result_xxxMeasurementRawData_output_m01_i01	References the ID in XML file of result for measurement raw data used as output

9.9 Naming <pnml>, <transition>, and <transitionRef> Elements

A process series can have a model for one series of processes indicated in a <pnml> element that is a child element of a <method> element. If there are multiple process series, multiple <pnml> elements are indicated. In this example, multiple <pnml> elements are indicated based on the <program> element.

Each <instruction> element corresponds to one Petri net transition and is indicated in a <transition> element that is a child element of a <pnml> element, with the ref attribute of the <transitionRef> element that is a child element of an <instruction> element referencing the id attribute of the <transition> element.

At this point, the <pnml>, <transition>, and <transitionRef> elements in the MaiML file where XXX measurement/analysis method data is stored can be named as indicated below, for example.

Table 9.8 Examples of Naming <pnml>, <transition>, and <transitionRef> Elements

<pnml> element		
XPath	/maiml/protocol/method[1]/pnml[1]	
<description> element	XXX Preparation Petri Net	Description of pnml for sample preparation
<name> element	xxx:PreparationPetriNet	Name of pnml for sample preparation (XML modified name)
id attribute	pnml_xxxPreparation	ID in XML file of pnml for sample preparation
<pnml> element		

XPath	/maiml/protocol/method[1]/pnml[2]	
<description> element	XXX Measurement PetriNet	Description of pnml for measurement
<name> element	xxx:MeasurementPetriNet	Name of pnml for measurement (XML modified name)
id attribute	pnml_xxxMeasurement	ID in XML file of pnml for measurement
<pnml> element		
XPath	/maiml/protocol/method[1]/pnml[3]	
<description> element	XXX Analysis Petri Net	Description of pnml for analysis
<name> element	xxx:AnalysisPetriNet	Name of pnml for analysis (XML modified name)
id attribute	pnml_xxxAnalysis	ID in XML file of pnml for analysis
<transition> element		
XPath	/maiml/protocol/method[1]/pnml[1]/transition[1]	
<description> element	XXX Preparation Transition	Description of transition for sample preparation
<name> element	xxx:PreparationTransition	Name of transition for sample preparation (XML modified name)
id attribute	transition_xxxPreparation	ID in XML file of transition for sample preparation
<transition> element		
XPath	/maiml/protocol/method[1]/pnml[2]/transition[1]	
<description> element	XXX Measurement Transition	Description of transition for measurement
<name> element	xxx:MeasurementTransition	Name of transition for measurement (XML modified name)
id attribute	transition_xxxMeasurement	ID in XML file of transition for measurement
<transition> element		
XPath	/maiml/protocol/method[1]/pnml[3]/transition[1]	
<description> element	XXX Qualitative Analysis Transition	Description of transition for qualitative analysis
<name> element	xxx:QualitativeAnalysisTransition	Name of transition for qualitative analysis (XML modified name)
id attribute	transition_xxxQualitativeAnalysis	ID in XML file of transition for qualitative analysis
<transitionRef> element		
XPath	/maiml/protocol/method[1]/program[1]/instruction[1]/transitionRef[1]	
<description> element	XXX Preparation Transition Reference	Description of transitionRef for sample preparation
<name> element	xxx:PreparationTransitionReference	Name of transitionRef for sample preparation (XML modified name)
id attribute	transitionRef_xxxPreparation	ID in XML file of transitionRef for sample preparation
ref attribute	transition_xxxPreparation	References the ID in XML file of transition for sample preparation
<transitionRef> element		
XPath	/maiml/protocol/method[1]/program[2]/instruction[1]/transitionRef[1]	
<description> element	XXX Measurement Transition Reference	Description of transitionRef for measurement
<name> element	xxx:MeasurementTransitionReference	Name of transitionRef for measurement (XML modified name)
id attribute	transitionRef_xxxMeasurement	ID in XML file of transitionRef for measurement
ref attribute	transition_xxxMeasurement	References the ID in XML file of transition for measurement
<transitionRef> element		
XPath	/maiml/protocol/method[1]/program[3]/instruction[1]/transitionRef[1]	
<description>	XXX Qualitative Analysis Transition	Description of transitionRef for

element	Reference	qualitative analysis
<name> element	xxx:QualitativeAnalysisTransitionReference	Name of transitionRef for qualitative analysis (XML modified name)
id attribute	transitionRef_xxxQualitativeAnalysis	ID in XML file of transitionRef for qualitative analysis
ref attribute	transition_xxxQualitativeAnalysis	References the ID in XML file of transition for qualitative analysis

9.10 Naming <place> and <placeRef> Elements

Each template element on the input or output side corresponds to a place for resources (information) on the input or output side of a Petri net and is indicated in a <place> element that is a child element of a <pnml> element, where the ref attribute for the <placeRef> element references the id attribute of the <place> element.

If there are multiple process series, multiple <pnml> elements are also indicated. Therefore, for this example, <place> elements are required for the template element used as an input for the operation in the previous process and for the template element copy used as the input for the operation in the subsequent process.

Note 1 Because the template element and copy template element both have input and output" sides, this example shows _output" or "_input" appended to the id attributes of corresponding <place> and <placeRef> elements.

Note 2 The same <description> and <name> elements were used for the <place> and <placeRef> elements corresponding to template and template copy elements (but with different id attributes).

At this point, the <place> and <placeRef> elements used to store XXX measurement/analysis method data in MaiML files can be named as indicated below, for example.

Table 9.9 Examples of Naming <place> and <placeRef> Elements

<place> element		
XPath	/maiml/protocol/method[1]/pnml[1]/place[1]	
<description> element	XXX Analyte Place	Description of place for analytes
<name> element	xxx:AnalytePlace	Name of place for analytes (XML modified name)
id attribute	place_xxxAnalyte_input	ID in XML file of place for analytes used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[1]/place[2]	
<description> element	XXX Preparation Settings Place	Description of place for sample preparation condition settings
<name> element	xxx:PreparationSettingsPlace	Name of place for sample preparation condition settings (XML modified name)
id attribute	place_xxxPreparationSettings_input	ID in XML file of place for sample preparation condition settings used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[1]/place[3]	
<description> element	XXX Sample Place	Description of place for sample
<name> element	xxx:SamplePlace	Name of place for sample (XML modified name)
id attribute	place_xxxSample_output	ID in XML file of place for sample used as output
<place> element		
XPath	/maiml/protocol/method[1]/pnml[2]/place[1]	
<description> element	XXX Sample Place	Description of place for sample
<name> element	xxx:SamplePlace	Name of place for sample (XML

		modified name)
id attribute	place_xxxSample_input	ID in XML file of place for sample used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[2]/place[2]	
<description> element	XXX Measurement Settings Place	Description of place for measurement condition settings
<name> element	xxx:MeasurementSettingsPlace	Name of place for measurement condition settings (XML modified name)
id attribute	place_xxxMeasurementSettings_input	ID in XML file of place for measurement condition settings used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[2]/place[3]	
<description> element	XXX Measurement Raw Data Place	Description of place for measurement raw data
<name> element	xxx:MeasurementRawDataPlace	Name of place for measurement raw data (XML modified name)
id attribute	place_xxxMeasurementRawData_output	ID in XML file of place for measurement raw data used as output
<place> element		
XPath	/maiml/protocol/method[1]/pnml[3]/place[1]	
<description> element	XXX Measurement Raw Data Place	Description of place for measurement raw data
<name> element	xxx:MeasurementRawDataPlace	Name of place for measurement raw data (XML modified name)
id attribute	place_xxxMeasurement RawData_input	ID in XML file of place for measurement raw data used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[3]/place[2]	
<description> element	XXX Qualitative Analysis Settings Place	Description of place for qualitative analysis condition settings
<name> element	xxx:QualitativeAnalysisSettingsPlace	Name of place for qualitative analysis condition settings (XML modified name)
id attribute	place_xxxQualitativeAnalysisSettings_input	ID in XML file of place for qualitative analysis condition settings used as input
<place> element		
XPath	/maiml/protocol/method[1]/pnml[3]/place[3]	
<description> element	XXX Qualitative Analysis Result Table Place	Description of place for qualitative analysis results table
<name> element	xxx:QualitativeAnalysisResultTablePlace	Name of place for qualitative analysis results table (XML modified name)
id attribute	place_xxxQualitativeAnalysisResultTable_output	ID in XML file of place for qualitative analysis results table used as output
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[1]/materialTemplate[1]/placeRef[1]	
<description> element	XXX Analyte Place Reference	Description of placeRef for analytes
<name> element	xxx:AnalytePlaceReference	Name of placeRef for analytes (XML modified name)
id attribute	placeRef_xxxAnalyte_input	ID in XML file of placeRef for analytes used as input
ref attribute	place_xxxAnalyte_input	References the ID in XML file of place for analytes used as input
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[1]/conditionTemplate[1]/placeRef[1]	
<description> element	XXX Preparation Settings Place Reference	Description of placeRef for sample preparation condition settings
<name> element	xxx:PreparationSettingsPlaceReference	Name of placeRef for sample preparation condition settings (XML modified name)

id attribute	placeRef_xxxPreparationSettings_input	ID in XML file of placeRef for sample preparation condition settings used as input
ref attribute	place_xxxPreparationSettings_input	References the ID in XML file of place for sample preparation condition settings used as input
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[1]/materialTemplate[2]/placeRef[1]	
<description> element	XXX Sample Place Reference	Description of placeRef for sample
<name> element	xxx:SamplePlaceReference	Name of placeRef for sample (XML modified name)
id attribute	placeRef_xxxSample_output	ID in XML file of placeRef for sample used as output
ref attribute	place_xxxSample_output	References the ID in XML file of place for sample used as output
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[2]/materialTemplate[2]/placeRef[1]	
<description> element	XXX Sample Place Reference	Description of placeRef for sample
<name> element	xxx:SamplePlaceReference	Name of placeRef for sample (XML modified name)
id attribute	placeRef_xxxSample_input	ID in XML file of placeRef for sample used as input
ref attribute	place_xxxSample_input	References the ID in XML file of place for sample used as input
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[2]/conditionTemplate[1]/placeRef[1]	
<description> element	XXX Measurement Settings Place Reference	Description of placeRef for measurement condition settings
<name> element	xxx:MeasurementSettingsPlaceReference	Name of placeRef for measurement condition settings (XML modified name)
id attribute	placeRef_xxxMeasurementSettings_input	ID in XML file of placeRef for measurement condition settings used as input
ref attribute	place_xxxMeasurementSettings_input	References the ID in XML file of place for measurement condition settings used as input
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[2]/resultTemplate[1]/placeRef[1]	
<description> element	XXX Measurement Raw Data Place Reference	Description of placeRef for measurement raw data
<name> element	xxx:MeasurementRawDataPlaceReference	Name of placeRef for measurement raw data (XML modified name)
id attribute	placeRef_xxxMeasurementRawData_output	ID in XML file of placeRef for measurement raw data used as output
ref attribute	place_xxxMeasurementRawData_output	References the ID in XML file of place for measurement raw data used as output
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[3]/resultTemplate[1]/placeRef[1]	
<description> element	XXX Measurement Raw Data Place Reference	Description of placeRef for measurement raw data
<name> element	xxx:MeasurementRawDataPlaceReference	Name of placeRef for measurement raw data (XML modified name)
id attribute	placeRef_xxxMeasurementRawData_input	ID in XML file of placeRef for measurement raw data used as input
ref attribute	place_xxxMeasurementRawData_input	References the ID in XML file of place for measurement raw data used as input
<placeRef> element		

XPath	/maiml/protocol/method[1]/program[3]/conditionTemplate[1]/placeRef[1]	
<description> element	XXX Qualitative Analysis Settings Place Reference	Description of placeRef for qualitative analysis condition settings
<name> element	xxx:QualitativeAnalysisSettingsPlaceReference	Name of placeRef for qualitative analysis condition settings (XML modified name)
id attribute	placeRef_xxxQualitativeAnalysisSettings_input	ID in XML file of placeRef for qualitative analysis condition settings used as input
ref attribute	place_xxxQualitativeAnalysisSettings_input	References the ID in XML file of place for qualitative analysis condition settings used as input
<placeRef> element		
XPath	/maiml/protocol/method[1]/program[3]/resultTemplate[2]/placeRef[1]	
<description> element	XXX Qualitative Analysis Result Table Place Reference	Description of placeRef for qualitative analysis results table
<name> element	xxx:QualitativeAnalysisResultTablePlaceReference	Name of placeRef for qualitative analysis results table (XML modified name)
id attribute	placeRef_xxxQualitativeAnalysisResultTable_output	ID in XML file of placeRef for qualitative analysis results table used as output
ref attribute	place_xxxQualitativeAnalysisResultTable_output	References the ID in XML file of place for qualitative analysis results table used as output

9.11 Naming <arc> Elements

Places on the input side are connected by arcs that indicate transition inputs, which are indicated in <arc> elements that are child elements of <pnml> elements, with a source attribute that references the id attribute of the <place> element and a target attribute that references the id attribute of the <transition> element.

Places on output side are connected by arcs that indicate transition outputs, which are indicated in <arc> elements that are child elements of <pnml> elements, with a source attribute that references the id attribute of the <transition> element and a target attribute that references the id attribute of the <place> element.

Note 1 Because the arcs have input and output sides, this example shows "_output" or "_input" appended to the id attributes of <arc> elements.

Note 2 Because arcs have an orientation, “Output" or "Input” is included in <description> and <name> elements of <arc> elements in this example.

At this point, the <arc> elements used to store XXX measurement/analysis method data in the MaiML file can be named as indicated below, for example.

Table 9.10 Examples of Naming <arc> Elements

<arc> element		
XPath	/maiml/protocol/method[1]/pnml[1]/arc[1]	
<description> element	XXX Analyte Input Arc	Description of arc for analyte used as input
<name> element	xxx:AnalyteInputArc	Name of arc for analyte used as input (XML modified name)
id attribute	arc_xxxAnalyte_input	ID in XML file of arc for analyte used as input
source attribute	place_xxxAnalyte_input	References the ID in XML file of place for analytes used as input
target attribute	transition_xxxPreparation	References the ID in XML file of transition for sample preparation
<arc> element		

XPath	/maiml/protocol/method[1]/pnml[1]/arc[2]	
<description> element	XXX Preparation Settings Input Arc	Description of arc for sample preparation condition settings used as input
<name> element	xxx:PreparationSettingsInputArc	Name of arc for sample preparation condition settings used as input (XML modified name)
id attribute	arc_xxxPreparationSettings_input	ID in XML file of arc for sample preparation condition settings used as input
source attribute	place_xxxPreparationSettings_input	References the ID in XML file of place for sample preparation condition settings used as input
target attribute	transition_xxxPreparation	References the ID in XML file of transition for sample preparation
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[1]/arc[3]	
<description> element	XXX Sample Output Arc	Description of arc for sample used as output
<name> element	xxx:SampleOutputArc	Name of arc for sample used as output (XML modified name)
id attribute	arc_xxxSample_output	ID in XML file of arc for sample used as output
source attribute	transition_xxxPreparation	References the ID in XML file of transition for sample preparation
target attribute	place_xxxSample_output	References the ID in XML file of place for sample used as output
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[2]/arc[1]	
<description> element	XXX Sample Input Arc	Description of arc for sample used as input
<name> element	xxx:SampleInputArc	Name of arc for sample used as input (XML modified name)
id attribute	arc_xxxSample_input	ID in XML file of arc for sample used as input
source attribute	place_xxxSample_input	References the ID in XML file of place for sample used as input
target attribute	transition_xxxMeasurement	References the ID in XML file of transition for measurement
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[2]/arc[2]	
<description> element	XXX Measurement Settings Input Arc	Description of arc for measurement condition settings used as input
<name> element	xxx:MeasurementSettingsInputArc	Name of arc for measurement condition settings used as input (XML modified name)
id attribute	arc_xxxMeasurementSettings_input	ID in XML file of arc for measurement condition settings used as input
source attribute	place_xxxMeasurementSettings_input	References the ID in XML file of place for measurement condition settings used as input
target attribute	transition_xxxMeasurement	References the ID in XML file of transition for measurement
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[2]/arc[3]	
<description> element	XXX Measurement Raw Data Output Arc	Description of arc for measurement raw data used as output
<name> element	xxx:MeasurementRawDataOutputArc	Name of arc for measurement raw data used as output (XML modified

		name)
id attribute	arc_xxxMeasurementRawData_output	ID in XML file of arc for measurement raw data used as output
source attribute	transition_xxxMeasurement	References the ID in XML file of transition for measurement
target attribute	place_xxxMeasurementRawData_output	References the ID in XML file of place for measurement raw data used as output
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[3]/arc[1]	
<description> element	XXX Measurement Raw Data Input Arc	Description of arc for measurement raw data used as input
<name> element	xxx:MeasurementRawDataInputArc	Name of arc for measurement raw data used as input (XML modified name)
id attribute	arc_xxxMeasurementRawData_input	ID in XML file of arc for measurement raw data used as input
source attribute	place_xxxMeasurementRawData_input	References the ID in XML file of place for measurement raw data used as input
target attribute	transition_xxxQualitativeAnalysis	References the ID in XML file of transition for qualitative analysis
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[3]/arc[2]	
<description> element	XXX Qualitative Analysis Input Arc	Description of arc for qualitative analysis condition settings used as input
<name> element	xxx:QualitativeAnalysisInputArc	Name of arc for qualitative analysis condition settings used as input (XML modified name)
id attribute	arc_xxxQualitativeAnalysis_input	ID in XML file of arc for qualitative analysis condition settings used as input
source attribute	place_xxxQualitativeAnalysis_input	References the ID in XML file of place for qualitative analysis condition settings used as input
target attribute	transition_xxxQualitativeAnalysis	References the ID in XML file of transition for qualitative analysis
<arc> element		
XPath	/maiml/protocol/method[1]/pnml[3]/arc[3]	
<description> element	XXX Qualitative Analysis Result Table Input Arc	Description of arc for qualitative analysis results table used as output
<name> element	xxx:QualitativeAnalysisResultTableInputArc	Name of arc for qualitative analysis results table used as output (XML modified name)
id attribute	arc_xxxQualitativeAnalysisResultTable_input	ID in XML file ofarc for qualitative analysis results table used as output
source attribute	transition_xxxQualitativeAnalysis	References the ID in XML file of transition for qualitative analysis
target attribute	place_xxxQualitativeAnalysisResultTable_output	References the ID in XML file of place for qualitative analysis results table used as output

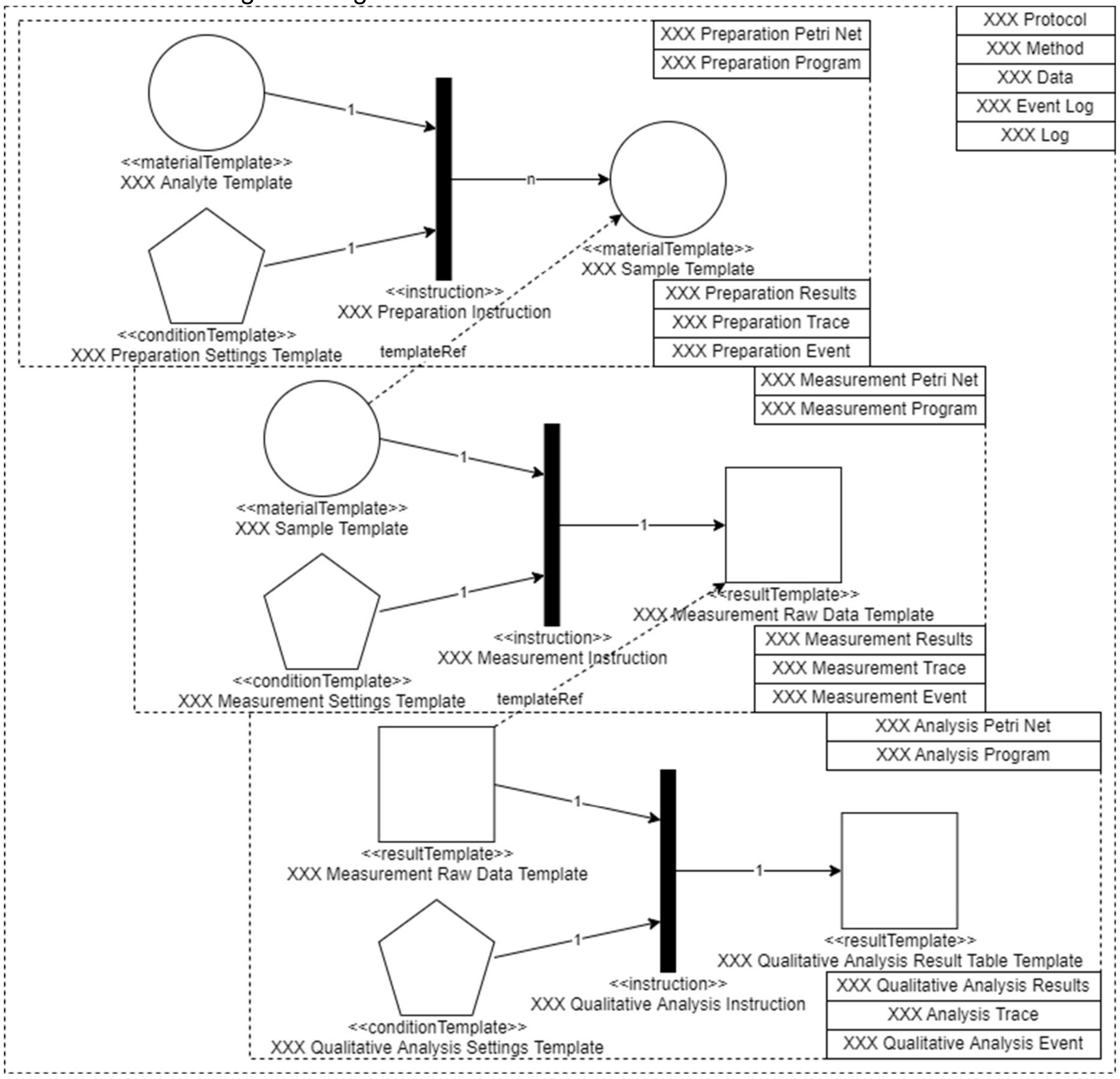
9.12 Examples of Measurement/Analysis Protocol Execution and Indicating Transition States for Event Log Recording Operations

A wide variety of cases are anticipated for recording logs from protocol execution results. In particular, MaiML files used for recording logs for laboratories or other applications, such as scheduling, starting, or completing processes normally or abnormally, are expected to offer utility for a variety of uses.

The following uses Petri net tokens to describe how MaiML file logs relate to protocol execution status.

9.12.1 Initial Protocol Execution Status

Use cases for event log recording methods are described below.



9.3 Initial Protocol Execution Status of XXX Measurement/Analysis Method

In Petri nets, transitions will not fire unless resource (information) tokens are located at a resource (information) place. Therefore, in the initial state before an XXX measurement/analysis method protocol is executed, there are no instance elements (which correspond to tokens) and no event logs are recorded, as shown in **Fig. 9.3**.

9.12.2 Example of Protocol and Experimental Design (Schedule)

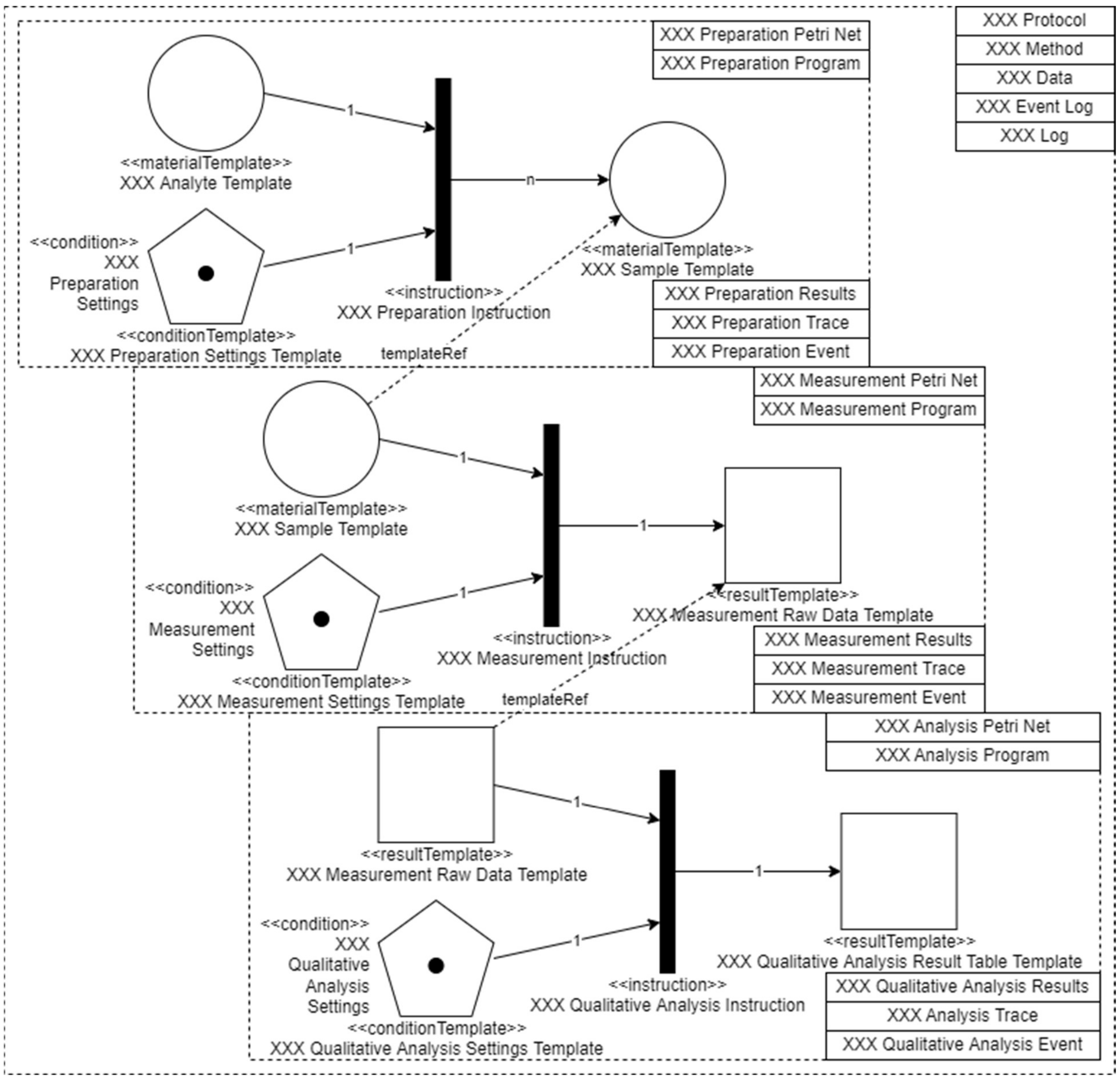


Fig. 9.4 Example of Protocol and Experimental Design (Schedule) for XXX Measurement/Analysis Method

This use case shows examples of modifying the example names for <event> or <resultsRef> elements in **Table 9.5** and examples for naming <event> or <resultsRef> elements corresponding to “schedule” and “start” status transition types, in addition to the "complete” status transition shown in **Fig. 9.2**.

In this case, only designated sample preparation and measurement operators prepare and measure samples, but after they obtain measurement results, they are also expected to perform the qualitative analysis themselves, using only periodically sampled specimens taken from the production line and stored in the plant warehouse. It is also assumed that condition settings for sample preparation, measurement, and qualitative analysis were decided in advance.

When the analyst receives notification that the specimens are stored in the plant warehouse, they create a work instructions sheet and, together with a list of sample preparation and measurement conditions, ask the sample preparation and measurement operators to execute the work instructions (**Fig. 9.4**).

Note 1 For this example, the “schedule” type status transition shown in **Fig. 9.2** was used for work instructions for sample preparation, measurement, and qualitative analysis conditions. In addition, "_schedule" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "schedule" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element, and the UUID value was indicated in the <condition> element in the <value> element with key attribute "concept:instance."

Note 2 Because the <description> and <name> elements for the <event> and <resultsRef> elements are the same as indicated in **Table 9.5** of section **9.6**, part of each sample name was included in id attributes to enable distinguishing between ID values for positioning sample preparation conditions, positioning analytes, starting sample preparation, and completing sample preparation. Note that to avoid redundancy, "preparation" was not included because it is already included in <name> elements.

At this point, instead of the <event> or <resultsRef> elements indicated in **Table 9.5** in section **9.6**, an event log is recorded for "schedule," which specifies instructions for setting sample preparation conditions, measurement conditions, and qualitative analysis conditions, where the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.11 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[1]	
<description> element	XXX Preparation Event	Description of event for sample preparation
<name> element	xxx:PreparationEvent	Name of event for sample preparation (XML modified name)
id attribute	event_xxxPreparationSettings_schedule_i01	ID in XML file of event for sample preparation condition setting instructions
ref attribute	instruction_xxxPreparation	References the ID in XML file of instruction for sample preparation
<event> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[1]	
<description> element	XXX Measurement Event	Description of event for measurement
<name> element	xxx:MeasurementEvent	Name of event for measurement (XML modified name)
id attribute	event_xxxMeasurementSettings_schedule_i01	ID in XML file of event for measurement condition setting instructions
ref attribute	instruction_xxxMeasurement	References the ID in XML file of instruction for measurement
<event> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]	
<description> element	XXX Qualitative Analysis Event	Description of event for qualitative analysis
<name> element	xxx:QualitativeAnalysisEvent	Name of event for qualitative analysis (XML modified name)
id attribute	event_xxxQualitativeAnalysisSettings_schedule_i01	ID in XML file of event for qualitative analysis condition setting event
ref attribute	instruction_xxxQualitativeAnalysis	References the ID in XML file of instruction for qualitative analysis
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[1]/resultsRef[1]	

<description> element	XXX Preparation Results Reference	Description of resultsRef for sample preparation
<name> element	xxx:PreparationResultsReference	Name of resultsRef for sample preparation (XML modified name)
id attribute	resultsRef_xxxPreparationSettings_schedule_i01	ID in XML file of resultsRef for sample preparation conditions setting instruction
ref attribute	results_xxxPreparation_i01	References the ID in XML file of results for sample preparation
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[1]/resultsRef[1]	
<description> element	XXX Measurement Results Reference	Description of resultsRef for measurement
<name> element	xxx:MeasurementResultsReference	Name of resultsRef for measurement (XML modified name)
id attribute	resultsRef_xxxMeasurementSettings_schedule_i01	ID in XML file of resultsRef for measurement conditions setting instruction
ref attribute	results_xxxMeasurement_i01	References the ID in XML file of results for measurement
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]/ resultsRef[1]	
<description> element	XXX Qualitative Analysis Results Reference	Description of resultsRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisResultsReference	Name of resultsRef for qualitative analysis (XML modified name)
id attribute	resultsRef_xxxQualitativeAnalysisSettings_schedule_i01	ID in XML file of resultsRef for qualitative analysis conditions setting instruction
ref attribute	results_xxxQualitativeAnalysis_i01	References the ID in XML file of results for qualitative analysis

9.12.3 Examples of Preparing for and Starting Protocols and Sample Preparation

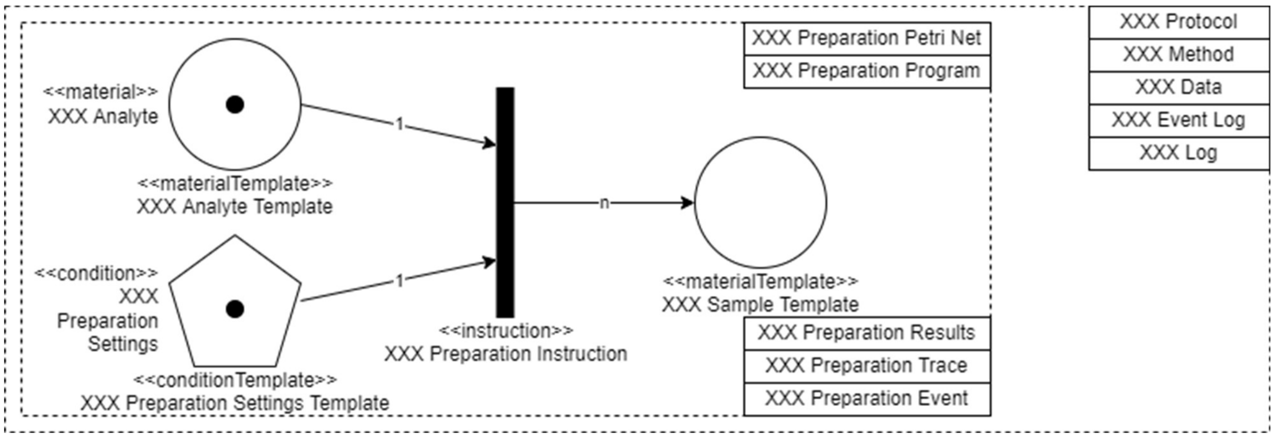


Fig. 9.5 Examples of Preparing for and Starting Protocols and Sample Preparation for XXX Measurement/Analysis Method

When the sample preparation operator receives notification that work instructions have been issued, they receive analytes from the plant warehouse, move to the sample preparation room, register the lot number in the sample management system, and check the sample preparation condition settings to complete the preparation for sample preparation (**Fig. 9.5**).

Note 1 For this example, the “schedule” type status transition shown in **Fig. 9.2** was used for sample preparation completion. In addition, "_schedule" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "schedule" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element, and the UUID value was indicated in the <material> element in the <value> element with key attribute "concept:instance."

Note 2 Because the <description> and <name> elements for the <event> and <resultsRef> elements are the same as indicated in **Table 9.5** of section **9.6**, part of each sample name was included in id attributes to enable distinguishing between ID values.

Note 3 This process step can also be skipped and included with the next "start" step.

After the event logs are recorded for instructions to specify settings for sample preparation conditions, measurement conditions, and qualitative analysis conditions, the "schedule" event log is recorded to indicate that preparation for preparing analyte samples is completed. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.12 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[2]	
<description> element	XXX Preparation Event	Description of event for sample preparation
<name> element	xxx:PreparationEvent	Name of event for sample preparation (XML modified name)
id attribute	event_xxxAnalyte_schedule_i01	ID in XML file of event for completion of preparation for preparing analyte samples
ref attribute	instruction_xxxPreparation	References the ID in XML file of instruction for sample preparation instruction
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[2]/resultsRef[1]	
<description> element	XXX Preparation Results Reference	Description of resultsRef for sample preparation
<name> element	xxx:PreparationResultsReference	Name of resultsRef for sample preparation (XML modified name)
id attribute	resultsRef_xxxAnalyte_schedule_i01	ID in XML file of resultsRef for completion of preparation for preparing analyte samples
ref attribute	results_xxxPreparation_i01	References the ID in XML file of results for sample preparation

When the sample preparation operator has finished preparation for preparing samples, they start actual sample preparation operations (**Fig. 9.5**).

Note 4 For this example, the "start" type status transition shown in **Fig. 9.2** was used for starting sample preparation. In addition, "_start" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "start" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element. Assuming the input instance element is immediately consumed when the process starts, indicating the "concept:instance" key attribute is not essential if the <resultsRef> element was specified.

Note 5 The <description> and <name> elements for the <event> and <resultsRef> elements are assumed to be the same as indicated in **Table 9.5** of section **9.6** (but with different id attributes).

After the analyte event log is recorded for completion of preparation for preparing analyte samples, the “start” event log is recorded to indicate the start of sample preparation. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.13 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[3]	
<description> element	XXX Preparation Event	Description of event for sample preparation
<name> element	xxx:PreparationEvent	Name of event for sample preparation (XML modified name)
id attribute	event_xxxPreparation_start_i01	ID in XML file of event for starting sample preparation
ref attribute	instruction_xxxPreparation	References the ID in XML file of instruction for sample preparation
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[3]/resultsRef[1]	
<description> element	XXX Preparation Results Reference	Description of resultsRef for sample preparation
<name> element	xxx:PreparationResultsReference	Name of resultsRef for sample preparation (XML modified name)
id attribute	resultsRef_xxxPreparation_start_i01	ID in XML file of resultsRef for starting sample preparation
ref attribute	results_xxxPreparation_i01	References the ID in XML file of results for sample preparation

9.12.4 Examples of Protocol and Sample Preparation Completion

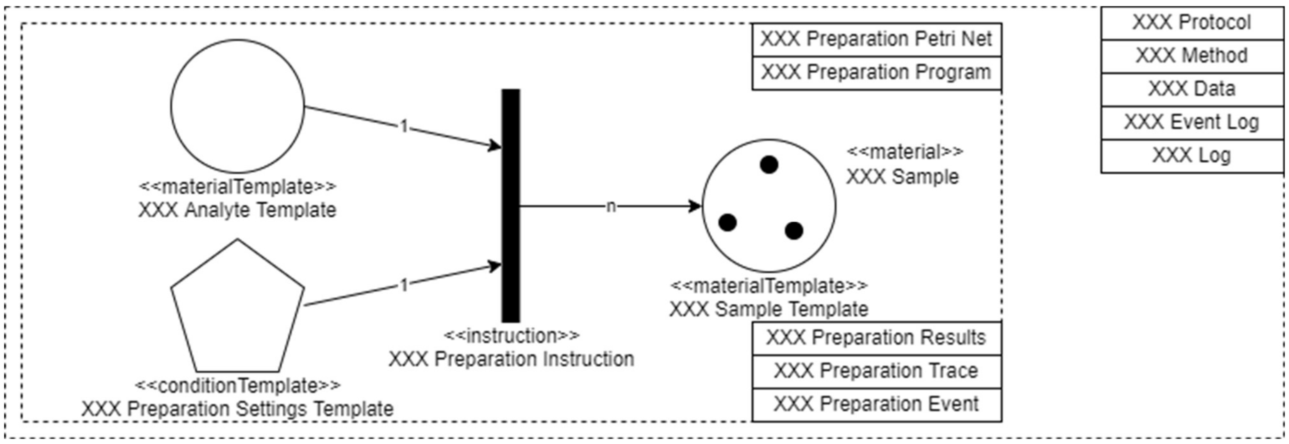


Fig. 9.6 Examples of Protocol and Sample Preparation Completion for XXX Measurement/Analysis Method

Next, when the sample preparation operator has finished preparing samples, they are considered finished when they prepare a sample preparation work report. Then they store the samples in the storage area of the sample preparation room (**Fig. 9.6**).

In Fig. 9.5, when tokens are located at the place on the input side and the transition is fired, Petri net rules specify that the number of tokens consumed is equal to the number of arcs on the input side (the value displayed on the arcs). Furthermore, when the transitions in **Fig. 9.6** are completed, the Petri net rules specify that the same number of tokens are generated as the number of arcs on the output side.

Note 1 For this example, the “complete” type status transition shown in **Fig. 9.2** was used for sample preparation completion. In addition, “_complete” was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, “complete” was indicated in the

<value> element with key attribute “lifecycle:transition” in the <property> element contained in the general purpose data container for the <event> element. When completed, it is not absolutely essential to indicate the <property> element for the key attribute “concept:instance” as long as the <resultsRef> element is specified, because the output instance elements are expected to be generated simultaneously.

Note 2 The <description> and <name> elements for the <event> and <resultsRef> elements are assumed to be the same as indicated in **Table 9.5** of section **9.6** (id attributes are also the same, but with a changed XPath).

After the event log for starting sample preparation is recorded, a “complete” event log is recorded to indicate completion of sample preparation. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.14 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[4]	
<description> element	XXX Preparation Event	Description of event for sample preparation
<name> element	xxx:PreparationEvent	Name of event for sample preparation (XML modified name)
id attribute	event_xxxPreparation_complete_i01	ID in XML file of event for sample preparation completion
ref attribute	instruction_xxxPreparation	References the ID in XML file of instruction for sample preparation
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/event[4]/resultsRef[1]	
<description> element	XXX Preparation Results Reference	Description of resultsRef for sample preparation
<name> element	xxx:PreparationResultsReference	Name of resultsRef for sample preparation (XML modified name)
id attribute	resultsRef_xxxPreparation_complete_i01	D in XML file of resultsRef for sample preparation completion
ref attribute	results_xxxPreparation_i01	References the ID in XML file of results for sample preparation

9.12.5 Examples of Preparing and Starting Protocols and Measurements

After samples are created by pretreatment, they are prepared for actual measurements as the next measurement/analysis operation, as described below.

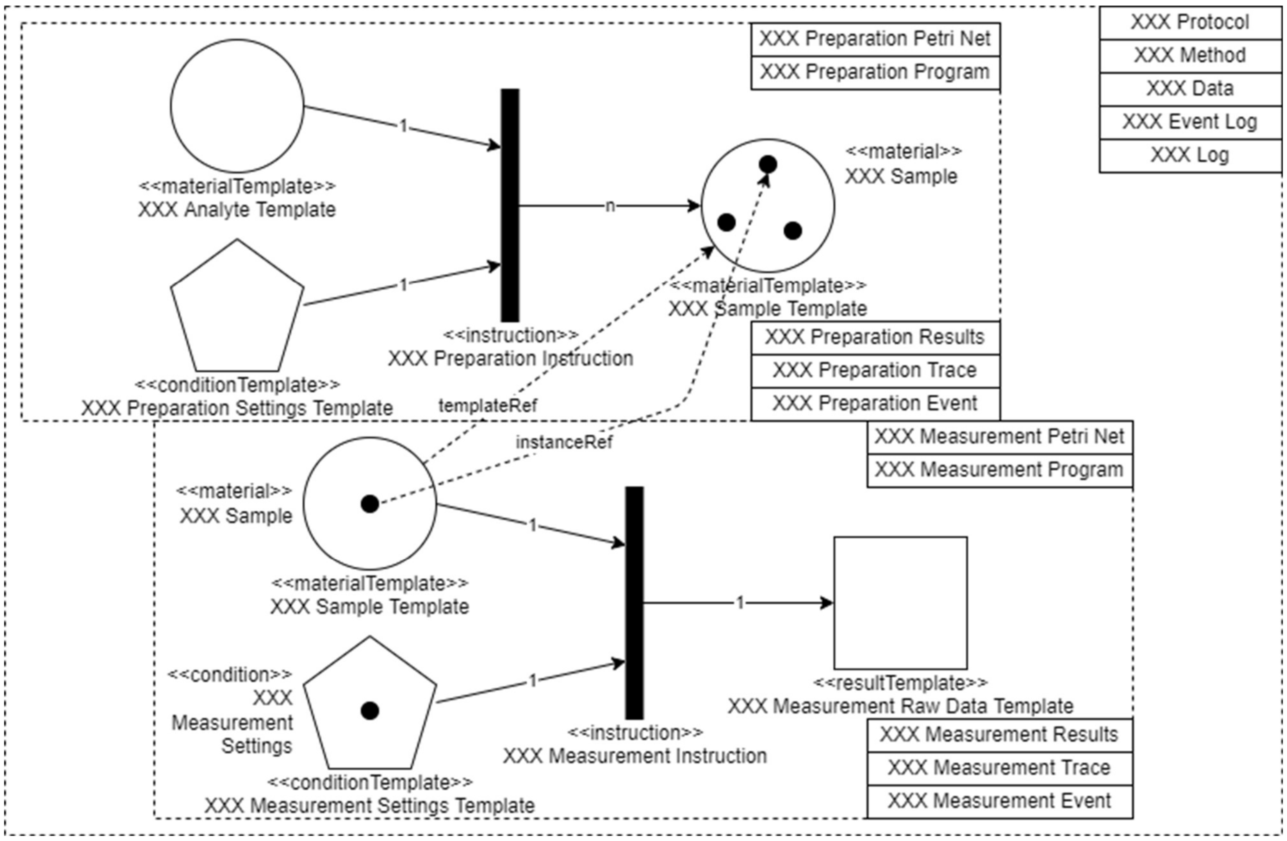


Fig. 9.7 Examples of Protocol and Measurement Preparation and Start for XXX Measurement/Analysis Method

When the measurement operator receives notification that a sample preparation work report has been issued, they receive samples from the storage area of the sample preparation room, move them to the sample measurement room, register the measurement schedule in the measurement/analysis system, and then check the measurement condition settings to complete the measurement preparation process (**Fig. 9.7**).

This example describes n = 3 tokens being generated by the output from the initial process, but only those initial tokens being processed during subsequent processes. In actuality, event logs are recorded for each sample, but MaiML files assume they are handled in separate files.

Note 1 For this example, the “schedule” type status transition shown in **Fig. 9.2** was used for measurement preparation completion. In addition, “_schedule” was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, “schedule” was indicated in the <value> element with key attribute “lifecycle:transition” in the <property> element contained in the general purpose data container for the <event> element, and the UUID value was indicated in the <material> element in the <value> element with key attribute “concept:instance.”

Note 2 Because the <description> and <name> elements for the <event> and <resultsRef> elements are the same as indicated in **Table 9.5** of section **9.6**, part of each sample name was included in id attributes to enable distinguishing between ID values for positioning measurement conditions, sample positioning, starting measurements, and completing measurements. Note that to avoid redundancy, “measurement” was not included because it is already included in <name> elements.

Note 3 This process step can also be skipped and included with the next "start” step.

After the event log for sample preparation completion is recorded, a “schedule” event log is recorded to indicate completion of preparation for sample measurements. At that point, the <event> and <resultsRef>

elements can be named as follows, for example.

Table 9.15 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[2]	
<description> element	XXX Measurement Event	Description of event for measurement
<name> element	xxx:MeasurementEvent	Name of event for measurement (XML modified name)
id attribute	event_xxxMeasurement_schedule_i01	ID in XML file of event for sample measurement preparation completion
ref attribute	instruction_xxxMeasurement	References the ID in XML file of instruction for measurement
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[2]/resultsRef[1]	
<description> element	XXX Measurement Results Reference	Description of resultsRef for measurement
<name> element	xxx:MeasurementResultsReference	Name of resultsRef for measurement (XML modified name)
id attribute	resultsRef_xxxSample_schedule_i01	ID in XML file of resultsRef for sample measurement preparation completion
ref attribute	results_xxxMeasurement_i01	References the ID in XML file of results for measurement

Next, when the measurement operator has finished preparing for measurements, they start actual measurement operations (**Fig. 9.7**).

Note 4 For this example, the “start” type status transition shown in **Fig. 9.2** was used for starting measurement. In addition, “_start” was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, “start” was indicated in the <value> element with key attribute “lifecycle:transition” in the <property> element contained in the general purpose data container for the <event> element. Assuming the input instance element is immediately consumed when the process starts, indicating the “concept:instance” key attribute is not essential if the <resultsRef> element was specified.

Note 5 The <description> and <name> elements for the <event> and <resultsRef> elements are assumed to be the same as indicated in **Table 9.5** of section **9.6** (but with different id attributes).

After the event log is recorded for completion of preparation for measuring samples, the “start” event log is recorded to indicate the start of measurement. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.16 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[3]	
<description> element	XXX Measurement Event	Description of event for measurement
<name> element	xxx:MeasurementEvent	Name of event for measurement event (XML modified name)
id attribute	event_xxxMeasurement_start_i01	ID in XML file of event for measurement start
ref attribute	instruction_xxxMeasurement	References the ID in XML file of instruction for measurement
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[3]/resultsRef[1]	
<description> element	XXX Measurement Results Reference	Description of resultsRef for measurement
<name> element	xxx:MeasurementResultsReference	Name of resultsRef for measurement (XML modified name)

id attribute	resultsRef_xxxMeasurement_start_i01	ID in XML file of resultsRef for measurement start
ref attribute	results_xxxMeasurement_i01	References the ID in XML file of results for measurement

9.12.6 Examples of Protocol and Measurement Completion

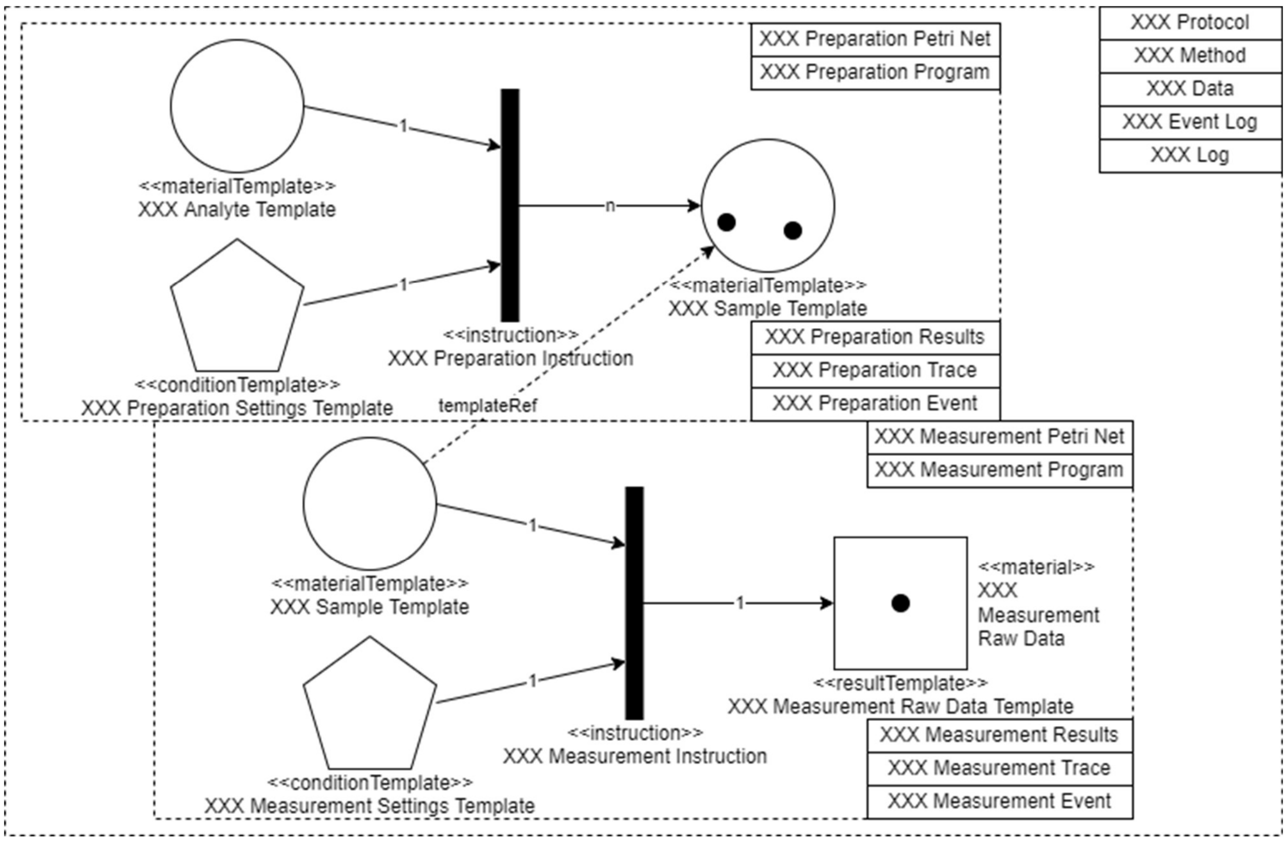


Fig. 9.8 Examples of Protocol and Measurement Completion for XXX Measurement/Analysis Method

Next, when the measurement operator has finished measurements, they are considered finished when they prepare a measurement work report. Then they store the measurement results on the server (**Fig. 9.8**).

Note 1 For this example, the “complete” type status transition shown in **Fig. 9.2** was used for measurement completion. In addition, “_complete” was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, “complete” was indicated in the <value> element with key attribute “lifecycle:transition” in the <property> element contained in the general purpose data container for the <event> element. When completed, it is not absolutely essential to indicate the <property> element for the key attribute “concept:instance” as long as the <resultsRef> element is specified, because the output instance elements are expected to be generated simultaneously.

Note 2 The <description> and <name> elements for the <event> and <resultsRef> elements are assumed to be the same as indicated in **Table 9.5** of section **9.6** (id attributes are also the same, but with a changed XPath).

After the event log for starting measurements is recorded, a “complete” event log is recorded to indicate measurement completion. At that point, the <event> and <resultsRef> elements can be named as follows, for example

Table 9.17 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[4]	
<description> element	XXX Measurement Event	Description of event for measurement
<name> element	xxx:MeasurementEvent	Name of event for measurement (XML modified name)
id attribute	event_xxxMeasurement_complete_i01	ID in XML file of event for measurement completion
ref attribute	instruction_xxxMeasurement	References the ID in XML file of instruction for measurement
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/event[4]/resultsRef[1]	
<description> element	XXX Measurement Results Reference	Description of resultsRef for measurement
<name> element	xxx:MeasurementResultsReference	Name of resultsRef for measurement (XML modified name)
id attribute	resultsRef_xxxMeasurement_complete_i01	ID in XML file of resultsRef for measurement completion
ref attribute	results_xxxMeasurement_i01	References the ID in XML file of results for measurement

9.12.7 Examples of Protocol and Qualitative Analysis Preparation and Start

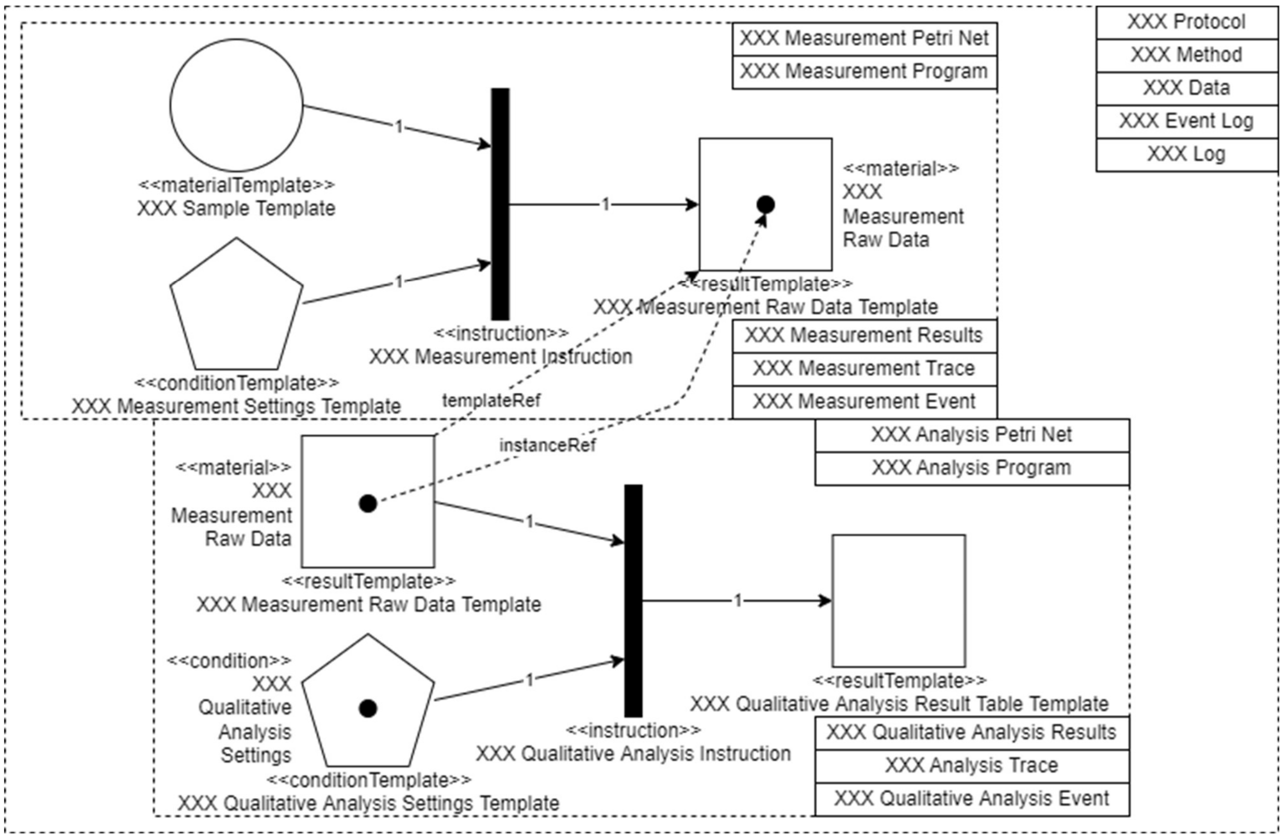


Fig. 9.9 Examples of Protocol and Qualitative Analysis Preparation and Start for XXX Measurement/Analysis Method

When the analyst receives notification that a measurement work report has been issued, they obtain the measurement results from the server, use the measurement/analysis system to prepare for qualitative analysis, and check qualitative analysis condition settings to complete the qualitative analysis preparation process (**Fig. 9.9**).

Note 1 For this example, the "schedule" type status transition shown in **Fig. 9.2** was used for completion of preparation for qualitative analysis. In addition, "_schedule" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "schedule" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element, and the UUID value was indicated in the <result> element for measurement raw data in the <value> element with key attribute "concept:instance."

Note 2 Because the <description> and <name> elements for the <event> and <resultsRef> elementsare the same as indicated in **Table 9.5** of section **9.6**, part of the measurement raw data name was included in id attributes to enable distinguishing between ID values for positioning qualitative analysis conditions, positioning measurement raw data, starting qualitative analysis, and completing qualitative analysis.

Note 3 This process step can also be skipped and included with the next "start" step.

After the event log for measurement completion is recorded, a "schedule" event log is recorded to indicate completion of preparation for qualitative analysis of measurement raw data. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.18 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[2]	
<description> element	XXX Qualitative Analysis Event	Description of event for qualitative analysis
<name> element	xxx:QualitativeAnalysisEvent	Name of event for qualitative analysis (XML modified name)
id attribute	event_xxxMeasurementRawData_schedule_i01	ID in XML file of event for completion of preparation for qualitative analysis of measurement raw data
ref attribute	instruction_xxxQualitativeAnalysis	References the ID in XML file of instruction for qualitative analysis
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[2]/ resultsRef[1]	
<description> element	XXX Qualitative Analysis Results Reference	Description of resultsRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisResultsReference	Name of resultsRef for qualitative analysis (XML modified name)
id attribute	resultsRef_xxxMeasurementRawData_schedule_i01	ID in XML file of resultsRef for completion of preparation for qualitative analysis of measurement raw data
ref attribute	results_xxxQualitativeAnalysis_i01	References the ID in XML file of results for qualitative analysis

Next, when the analyst name is finished preparing for qualitative analysis, they start actual qualitative analysis operations (**Fig. 9.9**).

Note 4 For this example, the "start" type status transition shown in **Fig. 9.2** was used for starting qualitative analysis. In addition, "_start" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "start" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element. Assuming the input instance element is immediately consumed when the process starts, indicating the "concept:instance" key attribute is not essential if the <resultsRef> element was specified.

Note 5 The <description> and <name> elements for the <event> and <resultsRef> elementsare assumed to be the same as indicated in **Table 9.5** of section **9.6** (but with different id attributes).

After the event log is recorded for completion of preparation for qualitative analysis of measurement raw data, the “start” event log is recorded to indicate the start of qualitative analysis. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.19 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[3]	
<description> element	XXX Qualitative Analysis Event	Description of event for qualitative analysis
<name> element	xxx:QualitativeAnalysisEvent	Name of event for qualitative analysis (XML modified name)
id attribute	event_xxxQualitativeAnalysis_start_i01	ID in XML file of event for start of qualitative analysis
ref attribute	instruction_xxxQualitativeAnalysis	References the ID in XML file of instruction for qualitative analysis
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[3]/ resultsRef[1]	
<description> element	XXX Qualitative Analysis Results Reference	Description of resultsRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisResultsReference	Name of resultsRef for qualitative analysis (XML modified name)
id attribute	resultsRef_xxxQualitativeAnalysis_start_i01	ID in XML file of resultsRef for start of qualitative analysis
ref attribute	results_xxxQualitativeAnalysis_i01	References the ID in XML file of results for qualitative analysis

9.12.8 Examples of Protocol and Qualitative Analysis Completion

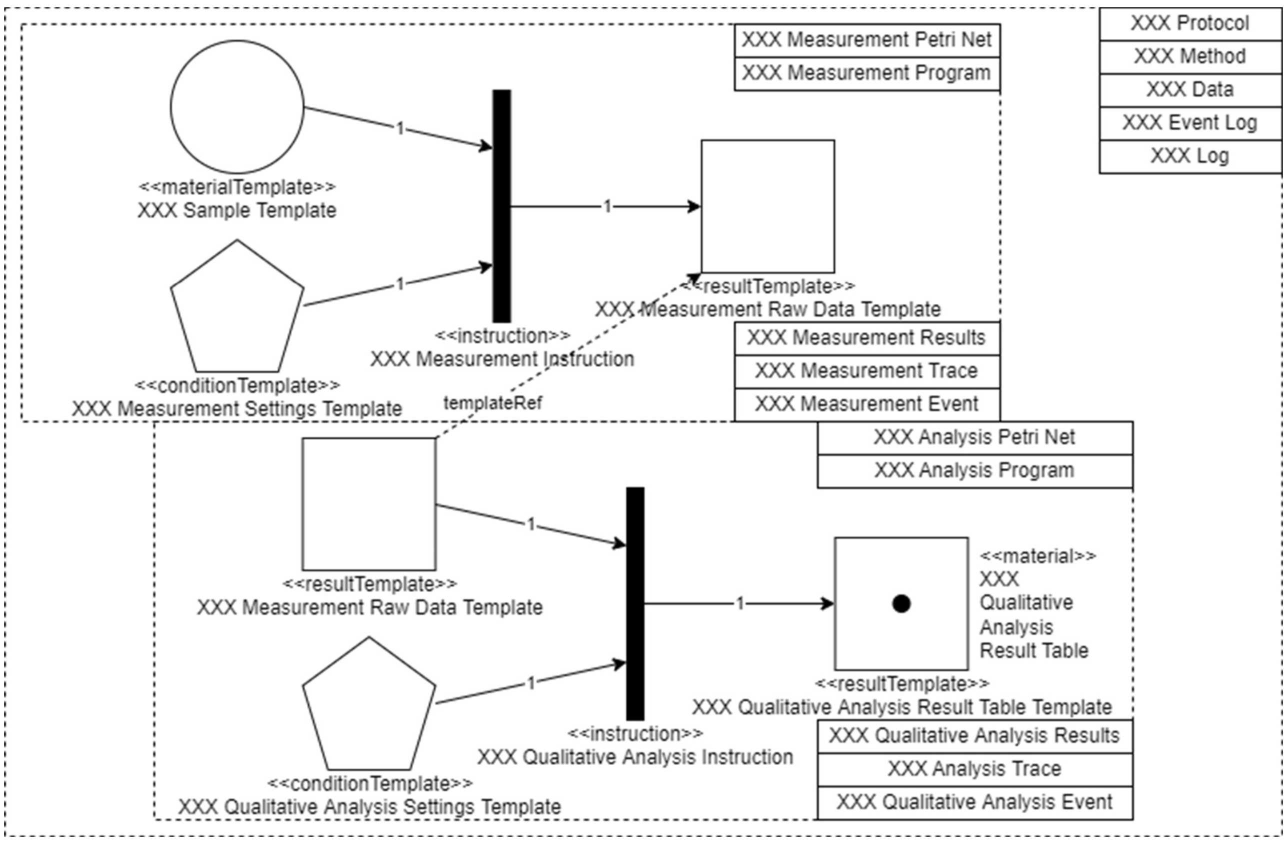


Fig. 9.10 Examples of Protocol and Qualitative Analysis Completion for XXX Measurement/Analysis Method

Next, when qualitative analysis is completed, the analyst completes the work by preparing a qualitative analysis report. Then they store the qualitative analysis results on the server (**Fig. 9.10**).

Note 1 For this example, the "complete" type status transition shown in **Fig. 9.2** was used for qualitative analysis completion. In addition, "_complete" was appended to the front of serial numbers in id attributes for <event> and <resultsRef> elements, "complete" was indicated in the <value> element with key attribute "lifecycle:transition" in the <property> element contained in the general purpose data container for the <event> element. When completed, it is not absolutely essential to indicate the <property> element for the key attribute "concept:instance" as long as the <resultsRef> element is specified, because the output instance elements are expected to be generated simultaneously.

Note 2 The <description> and <name> elements for the <event> and <resultsRef> elements are assumed to be the same as indicated in **Table 9.5** of section **9.6** (id attributes are also the same, but with a changed XPath).

After the event log for starting qualitative analysis recorded, a "complete" event log is recorded to indicate qualitative analysis completion. At that point, the <event> and <resultsRef> elements can be named as follows, for example.

Table 9.20 Examples of Naming <event> and <resultsRef> Elements

<event> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[4]	
<description> element	XXX Qualitative Analysis Event	Description of event for qualitative analysis
<name> element	xxx:QualitativeAnalysisEvent	Name of event for qualitative analysis (XML modified name)
id attribute	event_xxxQualitativeAnalysis_complete_i01	D in XML file of event for qualitative analysis completion
ref attribute	instruction_xxxQualitativeAnalysis	References the ID in XML file of instruction for qualitative analysis
<resultsRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[4]/ resultsRef[1]	
<description> element	XXX Qualitative Analysis Results Reference	Description of resultsRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisResultsReference	Name of resultsRef for qualitative analysis (XML modified name)
id attribute	resultsRef_xxxQualitativeAnalysis_complete_i01	ID in XML file of resultsRef for qualitative analysis completion
ref attribute	results_xxxQualitativeAnalysis_i01	References the ID in XML file of results for qualitative analysis

9.13 Measurement/Analysis Protocol Execution and Relationship between <owner> and <creator> Elements

9.13.1 Examples of Naming <owner> Elements

The <owner> element is a unique global element that is expected to use the <uuid> element to assign the same UUID value to the same individual or other unique entity. In this example, it refers to the XML modified name. If the name is combined with the namespace prefix as a value that ensures uniqueness and that can be expected to be written in the <name> element, then a UUID value can be generated from that <name> element value. However, in some cases that value is confidential and cannot be disclosed outside the organization. In such cases, specific individuals are identified by generating a UUID value using version 4, but that value should be kept separated from the <name> element.

The following describes a case where it may be possible to use a <name> element to distinguish confidentiality.

Example 1 Case with the Following Code between <name> and <uuid> Elements

```
xxx:PreparationOperator
xxx:MeasurementOperator
xxx:Analyst
```

In this case, each operation has at least one <event> element that is an operation log event and each <event> element corresponds to at least one <owner> element. However, for operations that comprise a series of processes, such as the sample preparation and measurement operation indicated in section **9.6**, one or more <owner> elements may correspond to the applicable <trace> element.
At this point, in order to distinguish between confidentiality, the <owner> or <ownerRef> elements in the MaiML file used to store XXX measurement/analysis method data can be named as indicated in **Table 9.21**.

Table 9.21 Examples of Naming <owner> and <ownerRef> Elements

<owner> element		
XPath	/maiml/document/owner[1]	
<description> element	XXX Preparation Operator	Description of owner for confidential sample preparation operator name
<name> element	xxx:PreparationOperator	Name of owner for confidential sample preparation operator name (XML modified name)
id attribute	owner_xxxPreparationOperator	ID in XML file of owner for confidential sample preparation operator name
<owner> element		
XPath	/maiml/document/owner[2]	
<description> element	XXX Measurement Operator	Description of owner for confidential measurement operator name
<name> element	xxx:MeasurementOperator	Name of owner for confidential measurement operator name (XML modified name)
id attribute	owner_xxxMeasurementOperator	ID in XML file of owner for confidential measurement operator name
<owner> element		
XPath	/maiml/document/owner[3]	
<description> element	XXX Analyst	Description of owner for confidential analyst name
<name> element	xxx:Analyst	Name of owner for confidential analyst name (XML modified name)
id attribute	owner_xxxAnalyst	ID in XML file of owner for confidential analyst name
<ownerRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/ownerRef[1]	
<description> element	XXX Preparation Owner Reference	Description of ownerRef for sample preparation
<name> element	xxx:PreparationOwnerReference	Name of ownerRef for sample preparation (XML modified name)
id attribute	ownerRef_xxxPreparation	ID in XML file of ownerRef for sample preparation
ref attribute	owner_xxxPreparationOperator	References the ID in XML file of owner for confidential sample preparation operator name
<ownerRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/ownerRef[1]	
<description> element	XXX Measurement Owner Reference	Description of ownerRef for measurement
<name> element	xxx:MeasurementOwnerReference	Name of ownerRef for measurement (XML modified name)

id attribute	ownerRef_xxxMeasurement	ID in XML file of ownerRef for measurement
ref attribute	owner_xxxMeasurementOperator	References the ID in XML file of owner for confidential measurement operator name
<ownerRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]/ownerRef[1]	
<description> element	XXX Qualitative Analysis Owner Reference	Description of ownerRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisOwnerReference	Name of ownerRef for qualitative analysis (XML modified name)
id attribute	ownerRef_xxxQualitativeAnalysis_complete_i01	ID in XML file of ownerRef for qualitative analysis
ref attribute	owner_xxxAnalyst	References the ID in XML file of owner for confidential analyst name

Note 1 The character string appended to the “ownerRef” prefix of id attributes for <ownerRef> elements are the same as for <trace> and <event> elements that are parent elements. In this case, the appended string indicates the corresponding series of processes or operations.

Note 2 In this case, the <event> element that is the parent element of the <ownerRef> element for qualitative analysis conforms to section **9.6** and the id attribute of the <ownerRef>, which is the child element of the <event> element, were appended with character strings “_complete” and “_i01” in conformance with the naming rule for id attributes for the <resultsRef> element specified in section **9.6**, **Notes 1** and **2**.

Note 3 If the <event> element that is the parent element of the <ownerRef> element for qualitative analysis conforms to section **9.12**, then the character string appended to the id attribute of the <ownerRef>, which is the child element of the <event> element, must conform with the naming rule for id attributes for the <resultsRef> element specified in section **9.12**.

According to personal information protection laws in respective countries, PIDs (personal identifiers) that are a part of the PII (personally identifiable information) used to identify individuals in the <name> element may be used as a provisional name within organizations but cannot be used as a confidential name disclosed outside organizations. Therefore, they should be embedded using a general purpose data container as data that can be deleted later, such as in the form “private:PIDxxxxxxxx,” “private:OperatorName,” or “private:eMail.”

Then an xmlns attribute can be used to indicate xmlns:private="http://www.example.com/owners#," as specified in **JIS K 0200**, **A.3**, to define the namespace as a prefix and use the modified name. If software used within an organization is exported outside the organization, it can be created as a new MaiML file with a <chain> element used to retain a link to that file, so that its relationship to the file is clearly indicated and changes to or tampering of the original file can be detected, for example.

If data contains ORCID or other personally identifiable information that can be openly disclosed, it may be disclosed in personally identifiable form, but it might be advisable to add “xxx:Analyst” or other data about additional roles (such as representatives or other additional owners). Also, when ORCID values are registered in the system, beware that some of the information might require the individual’s approval.

The following describes a case using an ORCID VPN. ORCID VPNs used URL values to ensure the uniqueness of data. An example of using URL values is indicated below.

```
<owner xmlns:orchid=”https://orchid.org/”>
   <uuid> 8d969f5c-4227-3a38-bfaf-defdfcbfa0f4 </uuid>
   <name>orchid:ORCHID0000-0002-2494-9603 </name>
</owner>
```

In addition, presumably employee numbers or other values used within an organization can also be used to specify <name> elements. In that case, the uniqueness of individuals cannot be guaranteed with employee numbers alone. In such cases, respective owner organizations can use their company name or other value as a URI for specifying the namespace and then generating UUID values. An example of that approach is given below.

Example 2 This is an example of indicating xmlns:pid="http://www.example.com/Employee/PersonalID#" to define the namespace as a prefix and using modified name with the PID linked to the personal name indicated in English. The characters "PID" were added to the beginning of PID values, because naming rules must conform to the guideline in Appendix B, which does not permit names that start with a number.

```
<owner xmlns:pid="http://www.example.com/Employee/PersonalID#">
   <uuid>b32175ea-e20c-3ce1-a1f1-91c111530d54</uuid>
   <name>pid:PID0233-T.Yasunaga</name>
</owner>
```

Because hash function calculations are not reversible, if UUID values are disclosed, the <name> element and xmlns attribute can be deleted to prevent values in <name> elements from being predicted based on the values in <uuid> elements. In addition, if Version 4 is used to randomize UUID values, then individuals must have a table in order to always use the same UUID value but, more importantly, it enables the management of UUID values.

9.13.2 Examples of Naming <creator> Elements

<creator> elements are also unique global elements used to specify specific instruments or software. They are expected to have a <uuid> element with a UUID value for specific devices or software. Though <uuid> element values can also be generated based on <name> elements that are an XML modified name, doing so requires being careful to keep in mind the specific devices when assigning names. Instruments with the same model number but a different serial number, version, or other characteristic are expected to be specified as a different instrument.

An example of using a namespace to name a <creator> element is indicated below.

```
xxx:SampleManagementSystem_1_0
(For "Preparation")
xxx:Model-XXX_2_51
xxx:XxxMeasurementAnalysisSoftware_1_0
(For "Measurement") xxx:XxxMeasurementAnalysisSoftware_1_1
(For "Analysis")
```

In this case, each operation has at least one <event> element that is an operation log and each <event> element corresponds to at least one <creator> element. However, for operations that comprise a series of processes, such as the sample preparation and measurement operation indicated in section **9.6**, one or more <creator> elements may correspond to the applicable <trace> element.

At this point, in order to distinguish between serial numbers and versions, the <creator> or <creatorRef> elements that store XXX measurement/analysis method data in the MaiML file can be named as indicated in **Table 9.22**.

Table 9.22 Examples of Naming <creator> and <creatorRef> Elements

<creator> element		
XPath	/maiml/document/creator[1]	
<description> element	Sample Management System, Version 1.0, S/N: M001-00001	Description of creator for sample management system
<name> element	xxx:SampleManagementSystem_M001-00001_1.0	Name of creator for sample management system (XML modified name)
id attribute	creator_xxxSampleManagementSystem	ID in XML file of creator in sample

		management system
<creator> element		
XPath	/maiml/document/creator[2]	
<description> element	XXX Measurement Hardware, Model-XXX, Version 2.51, S/N: H001-00001	Description of creator for measurement hardware
<name> element	xxx:Model-XXX_H001-00001_2.51	Name of creator for measurement hardware (XML modified name)
id attribute	creator_xxxMeasurementHardware	ID in XML file of creator for measurement hardware
<creator> element		
XPath	/maiml/document/creator[3]	
<description> element	XXX Measurement Analysis Software, Version 1.0, S/N: S001-00001	Description of creator for measurement software
<name> element	xxx:MeasurementAnalysisSoftware_S001-00001_1.0	Name of creator for measurement software (XML modified name)
id attribute	creator_xxxMeasurementSoftware	ID in XML file of creator for measurement software
<creator> element		
XPath	/maiml/document/creator[4]	
<description> element	XXX Measurement Analysis Software, Version 1.1, S/N: S001-00123	Description of creator for analysis software
<name> element	xxx:MeasurementAnalysisSoftware_S001-00123_1.1	Name of creator for analysis software (XML modified name)
id attribute	creator_xxxAnalysisSoftware	ID in XML file of creator for analysis software
<creatorRef> element		
XPath	/maiml/eventLog/log[1]/trace[1]/creatorRef[1]	
<description> element	XXX Preparation Creator Reference	Description of creatorRef for sample preparation
<name> element	xxx:PreparationCreatorReference	Name of creatorRef for sample preparation (XML modified name)
id attribute	creatorRef_xxxPreparation	ID in XML file of creatorRef for sample preparation
ref attribute	creator_xxxSampleManagementSystem	References the ID in XML file of creator for sample management system
<creatorRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/creatorRef[1]	
<description> element	XXX Measurement Creator Reference	Description of creatorRef for measurement
<name> element	xxx:MeasurementCreatorReference	Name of creatorRef for measurement (XML modified name)
id attribute	creatorRef_xxxMeasurement_m01	ID in XML file of creatorRef for measurement
ref attribute	creator_xxxMeasurementHardware	References the ID in XML file of creator for measurement hardware
<creatorRef> element		
XPath	/maiml/eventLog/log[1]/trace[2]/creatorRef[2]	
<description> element	XXX Measurement Creator Reference	Description of creatorRef for measurement
<name> element	xxx:MeasurementCreatorReference	Name of creatorRef for measurement (XML modified name)
id attribute	creatorRef_xxxMeasurement_m02	ID in XML file of creatorRef for measurement
ref attribute	creator_xxxMeasurementSoftware	References the ID in XML file of creator for measurement software
<creatorRef> element		
XPath	/maiml/eventLog/log[1]/trace[3]/event[1]/creatorRef[1]	
<description> element	XXX Qualitative Analysis Creator Reference	Description of creatorRef for qualitative analysis
<name> element	xxx:QualitativeAnalysisCreatorReference	Name of creatorRef for qualitative analysis (XML modified name)
id attribute	creatorRef_xxxQualitativeAnalysis_complete	ID in XML file of creatorRef for

	_i01	qualitative analysis
ref attribute	creator_xxxAnalysisSoftware	References the ID in XML file of creator for analysis software

Note 1 The instrument model, software version, serial number, and version number were included in the <name> element of <creator> elements. For a list of valid characters, refer to the guideline in **Appendix B**. This modified name can be uniquely named in the vendor namespace. Modified names that do not contain a serial number or version number can also be defined for confidential <creator> elements.

Note 2 The character string appended to the “creatorRef” prefix of id attributes for <creatorRef> elements are the same as for <trace> and <event> elements that are parent elements. In this case, the appended string indicates the corresponding series of processes or operations.

Note 3 It is assumed that hardware and software are indicated in the measurement <creator> element. The id attribute for the <creatorRef> element can be named based on the modified names for hardware and software, but because there may be more than one type (quantity) each, attributes were appended with “_m01” or “_m02” strings, where “m” refers to “module.”

Note 4 The <creator> element for the analysis software has the same package as for the measurement software, but different serial numbers and version numbers were assigned.

Note 5 In this case, the <event> element that is the parent element of the <creatorRef> for qualitative analysis conforms to section **9.6** and the id attribute of the <creatorRef>, which is the child element of the <event> element, were appended with character strings "_complete" and "_i01" in conformance with the naming rule for id attributes for the <resultsRef> element specified in section **9.6**, **Notes 1** and **2**.

Note 6 If the <event> element that is the parent element of the <creatorRef> element for qualitative analysis conforms to section **9.12**, then the character string appended to the id attribute of the <creatorRef>, which is the child element of the <event> element, must conform with the naming rule for id attributes for the <resultsRef> element specified in section **9.12**.

In this case, the dash number indicates the version or other information, with identical output expected from measurement/analysis of the same samples using the same conditions. Therefore, if details from each sample are required, additional information, such as xxx:SerialNumber or xxx:Version, is expected to be included in general purpose data containers.

In addition, presumably it would also be possible to specify <name> elements using serial numbers for respective instruments. In that case, uniqueness cannot be guaranteed for all analytical instruments using serial numbers alone. Therefore, the vendor of each instrument can use instrument name, version, or other information as URI values for specifying namespaces in order to generate UUID values.

Example In this example, xmlns: xxx="http://www.aVendor.co.jp/instrument/SEM-2000/1.2/SerialNumber#" defines the namespace as a prefix and uses the serial number as a modified name. As the naming rule, “SN” was added to the being of serial numbers in order to conform to the guideline in **Appendix B**.

```
<creator xmlns:xxx=”http://www.aVendor.co.jp/instrument/SEM-2000/1.2/SerialNumber#”>
   <uuid>2d222d4e-c606-3f81-abeb-356c62466e9c</uuid>
   <name>xxx:SN00120340-30T</name>
</creator>
```

That is because it is easier to use serial numbers as UUID values than random numbers generated by Version 4 that must be continuously retained for each instrument.

10. Measurement/Analysis Use Cases

An example of a use case related to measurement/analysis that involves indicating combined measurements is described in **10.1** based on using a GC-MS system. A use case involving combined measurements/analysis via cyberspace is described in **10.2**.

10.1 Example of Indicating Combined Measurements

The following describes a case of posting a combination of GC-MS measurements. The GC-MS system comprised two main modules based on different separation techniques, which were (1) a gas chromatograph (GC) for chromatographic separation and (2) a mass spectrometer (MS) for mass separation. The GC (gas chromatograph) unit separates the various chemical components contained in the mixture sample and outputs a series of corresponding peaks in the form of a chromatogram. Each component output as a set of corresponding peaks was then ionized by the MS (mass spectrometer) unit to analyze their respective masses.

Furthermore, the spectrum output from the MS unit can be used in combination with spectral library data to predict the corresponding molecules and the signal intensity can be used for quantitative analysis.

A Petri net of the series of measurement/analysis processes is shown below.

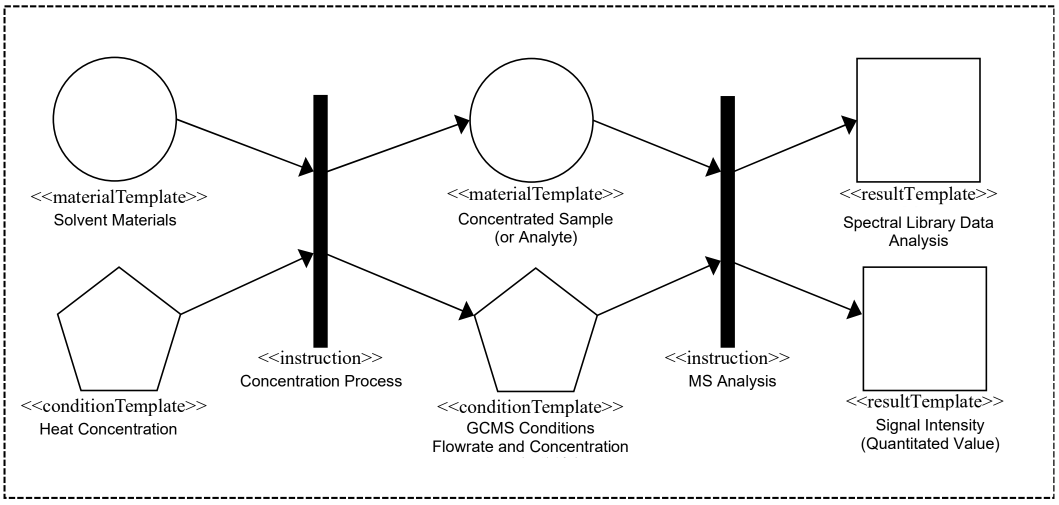


Fig. 10.1 Petri Net of the Series of GC-MS Measurement Processes

10.2 Use Cases Related to Using Data

10.2.1 Samples Created In-House, Measurements by Outside Companies, and Data Combined in Cyberspace

In this example, samples were prepared in-house and testing was outsourced to outside companies A and B.

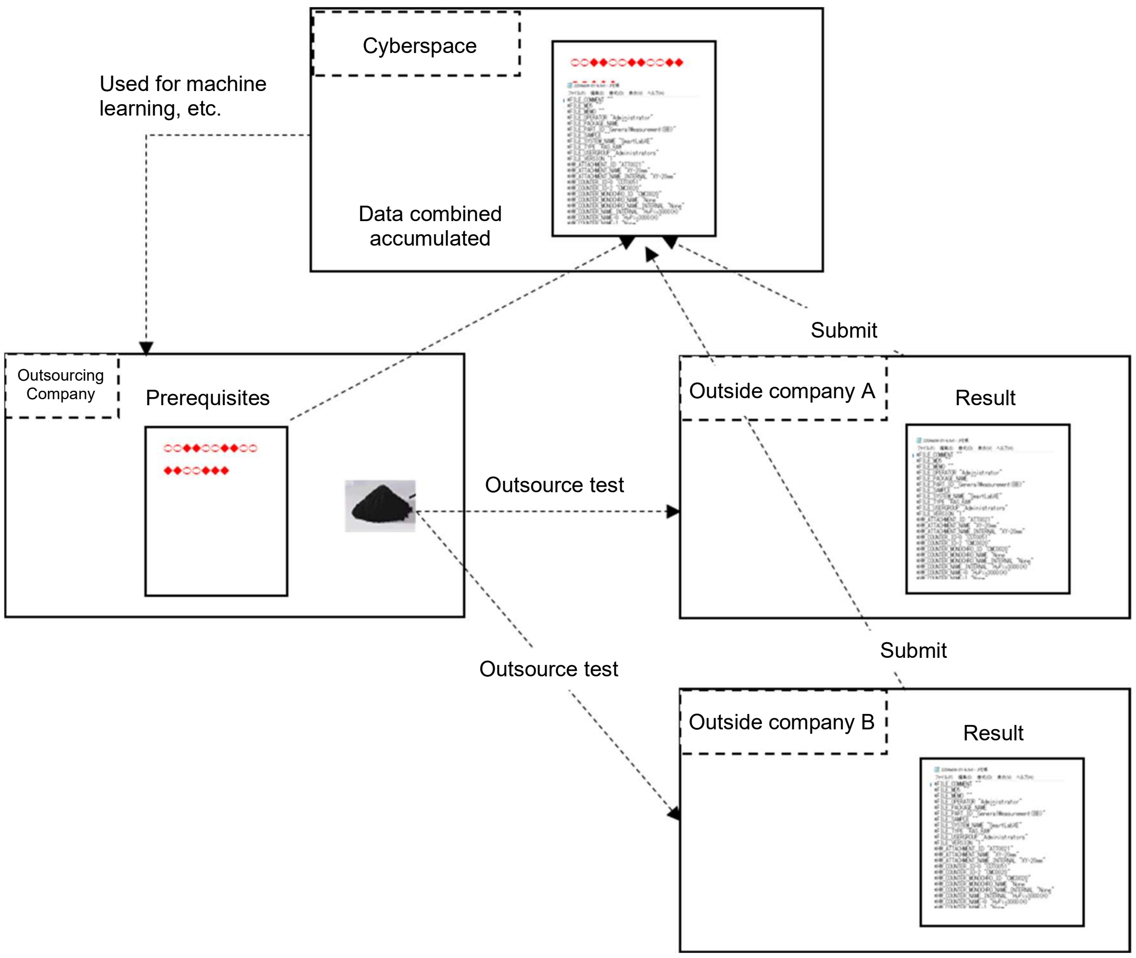


Fig. 10.2 Example of Outsourcing Testing to Outside Companies and Sharing Data in Cyberspace

At this point, by following the steps indicated below, the testing results from outside companies A and B can be submitted to cyberspace and combined without revealing sample details. One example is described below.

(1) The outsourcing company generates UUID values for each sample (Ver. 4) and creates MaiML files with <material> elements that contain the generated UUID values as descendant elements of a <data> element.

  (a) If a MaiML file that indicates information for processes extending up to sample creation already exists, then a new MaiML file is generated for outsourcing, which should be linked using a <chain> element. Either the same UUID value or a newly generated value may be used for that process. If newly generated, a Petri net and <instruction> elements are used to show the relationship between respective <material> elements.

  (b) If the outsourcing company has an in-house database or data warehouse, sample UUID values, <uuid> element values in <document> elements, or other information contained in the MaiML file can be added to the database/data warehouse.

(2) If testing is outsourced, a MaiML file that contains a <material> element with a UUID value created in step (1) is used for outsourcing testing to outside companies. The UUID values ensure samples are identical.

  (a) The minimum information required for inspections (such as concentration) can also be indicated in that MaiML file.

  (b) Information not to be disclosed to outside companies can also be removed from MaiML files.

(c) By attaching QR codes to the samples themselves, UUID values, values in <uuid> elements contained in MaiML file <document> elements, and so on, can be linked.

(3) When the outside organizations receive a testing request, they use the <results> element that is a descendant element of the <data> element prepared for each measurement and the <material> element that contains the same UUID values as provided for each sample to create a MaiML file with test results contained in a <result> element and then submit it to cyberspace.

(a) If pretreatment is required for testing, pretreatment information should be attached.

(b) If a MaiML file is attached when the request is received, keeping that file linked by a <chain> element will clearly indicate that relationship and enable data tampering to be detected.

(c) Information about testing condition settings that the outside companies do not want to provide can be deleted or confidentiality can be applied to the MaiML file before it is submitted to cyberspace.

1) If the <chain> elements are used to clearly indicate the links to MaiML files received from the outsourcing company, then it is easy to search for links.

2) If <chain> elements are also used to link MaiML files that contain confidential information stored within the outside companies, then when the outside companies are asked to provide those files the links can be used to ensure the outside companies did not tamper with that information.

(4) Test information in <material> elements with the same UUID value in MaiML files submitted to cyberspace by multiple outside testing companies can be combined or analyzed as test information from the same sample.

(a) If the respectively submitted MaiML files are linked by <chain> elements, then it is easy to search for the outsourcing company's original MaiML file to ensure the original MaiML file was not changed or tampered with.

(b) To treat the files as a new combined MaiML file, either all elements can be copied or only UUID values transferred.

(5) If data is to be analyzed at the outsourcing company, information in the MaiML file at the outsourcing company can be searched based on UUID values.

10.2.2 Sharing Data between Contractors (Including Using Data from Relevant Departments within a Company)

The following describes using a cyberspace on the Internet to share data between industry, academic, and corporate organizations or on the Intranet to share data between relevant departments within a company. If using the MaiML data format to evaluate or analyze a variety of documents, the necessary information can be obtained from files in a data lake located in a cyberspace.

In that case, if confidentiality has been applied to the necessary information, then a key for removing confidentiality can be given to another party so they can access the information.

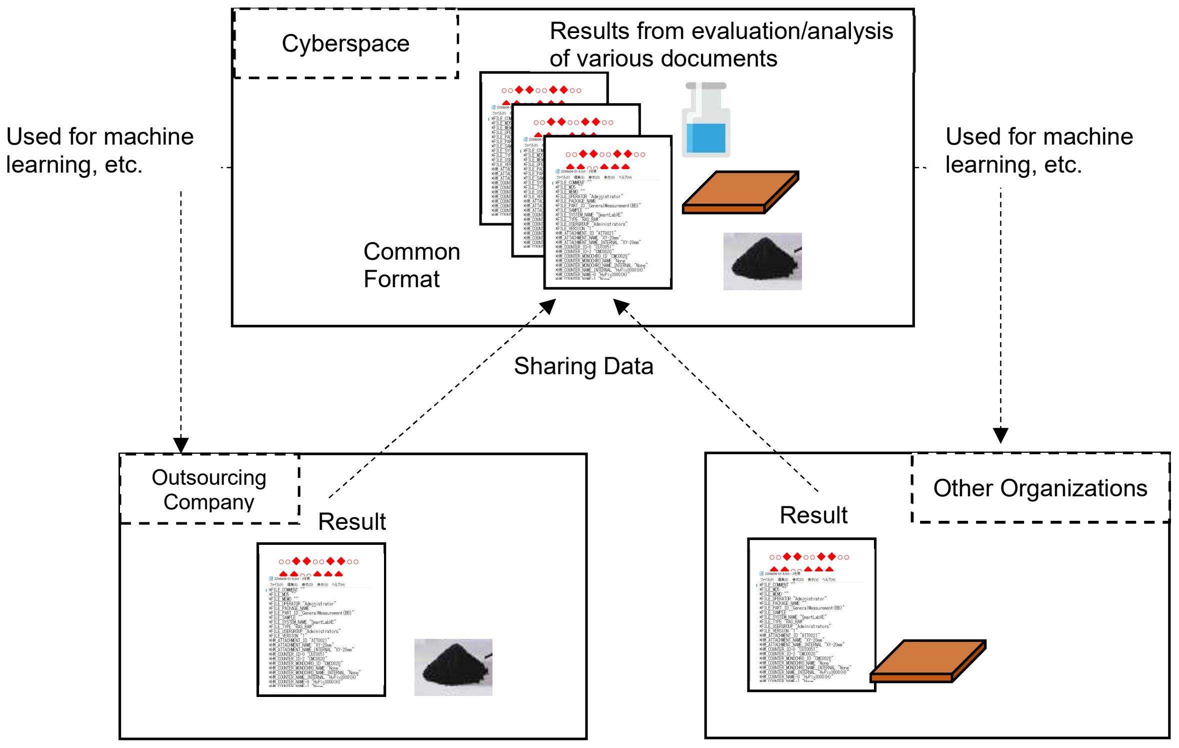

Cyberspace
Results from evaluation/analysis of various documents
Used for machine learning, etc.
Used for machine learning, etc.
Common Format
Sharing Data
Outsourcing Company
Result
Result
Other Organizations

Fig. 10.3 Examples of Using Cyberspace to Share Data between Industry, Academic, Government, or Corporate Organizations

At this point, necessary information intended for machine learning can be obtained from the MaiML files.

The next example describes using Python to convert information specified in a MaiML file into a CSV format file. At a stage during that process, the data expressed as tabular data can be used for many machine learning and deep learning uses.

Ex. 1 If the Internal XML Structure is Known

```
import csv
import glob
import xmltodict

maimlfiles = glob.glob('path/to/*')
fileNum = len(maimlfiles)
index = 0

HEADERS = ['key', 'description']
rows = []

namespaces = {
        'http://www.maiml.org/schemas': None,
        'http://www.example.com/maiml/material#' : 'exm' ,
}

# Read from maiml
for i in range(fileNum):
    maiml = maimlfiles[index]
    index += 1
    with open(maiml, 'r') as inF:
        maiml_dic = xmltodict.parse(inF.read(), process_namespaces=True,
          namespaces=namespaces)
        protocol_list = [dict(x) for x in maiml_dic['maiml']['protocol']['method']
                              ['program']['materialTemplate']['property']]

        # to row data
        for data in protocol_list:
            #print(data)
            key = data['@key']
            if 'description' in data.keys():
                description = data['description']
            else:
                description = ""
            rows.append([key, description])

# Write to CSV
with open(' path/to/file.csv', 'w', newline='') as outF:
    writer = csv.writer(outF)
    writer.writerow(HEADERS)
    writer.writerows(rows)
```

Original MaiML File

```
<?xml version="1.0" encoding="UTF-8"?>
<maiml version="1.0" features="nested-attributes"
            xmlns="http://www.maiml.org/schemas"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="rootObjectType">
    <document id="MaimlSampleDoc">
    </document>
    <protocol id="MaimlSampleProtocol">
        <uuid>5ac5ab10-866d-4ea2-b9f0-f9394d66b66c</uuid>
        <name>SEM_protpcol</name>
        <method id="methodID">
            <program id="programID">
                <materialTemplate id="MM_T">
                    <uuid>33e3c15c-81fb-4275-95f2-16504d0d16ce</uuid>
                    <name>material_XdashTemplate</name>
                    <property xsi:type="stringType" key="exm:MaterialLotNo">
                         <description>Lot No. of Material</description>
                        <value>Lot No.</value>
                        <property xsi:type="stringType" key="exm:LotNumber"></property>
                    </property>
                    <property xsi:type="stringType" key="exm:MaterialType">
                        <description>Type of Material</description>
                        <value> Bulk </value>
                        <property xsi:type="doubleType" key="exm:Thickness" formatString="0.00"
                                    scaleFactor="1.00" unit="mm">
                           <value> 10.00 </value>
                        </property>
                    </property>
                    <property xsi:type="stringType" key="exm:Comment">
                        <description>Comment for Material</description>
                        <value> "Color is black." </value>
                    </property>
                    <placeRef id="pr1" ref="MM-1"></placeRef>
                    <placeRef id="pr2" ref="MM-2"></placeRef>
                </materialTemplate>
                <conditionTemplate id="CC-1T">
                    <uuid>faa939ab-e634-4e33-99d0-bfd1545f4317</uuid>
                    <name>SEMconditionTemplate</name>
                    <description>SEM Condition</description>
                    <property xsi:type="doubleType" key="exm:AcceleratingVoltage" formatString="1.00"
```

```
                            units="kV" sacleFactor="1.0">
                        <value> 5.00 </value>
                    </property>
                    <property xsi:type="stringType" key="exm:Note">
                        <value> Low dose condition for redusing electron damage </value>
                    </property>
                    <placeRef id="pr4" ref="CC-1" ></placeRef>
                </conditionTemplate>
                <resultTemplate id="RR-2T">
                    <uuid>cc77bc35-267b-4d25-a76c-6b726a849028</uuid>
                    <name>SEMImageTemplate</name>
                    <property xsi:type="uriType" key="exm:SEMImageUri">
                    </property>
                    <property xsi:type="uriType" key="exm:SEMOutConditionUri">
                    </property>
                    <placeRef id="pr6" ref="RR-2" ></placeRef>
                </resultTemplate>
            </program>
        </method>
    </protocol>
</maiml>
```

Output CSV File

```
key,description
exm:MaterialLotNo,Lot No. of Material
exm:MaterialType,Type of Material
exm:Comment,Comment for Material
```

Ex. 2 If the Internal XML Structure is Unknown and Data is Obtained Using Key Attributes

```
import pandas as pd
import glob
import xml.etree.ElementTree as ET

maimlfiles = glob.glob(' path/to/*')
fileNum = len(maimlfiles)
index = 0
HEADERS = ['key', 'description']
row = []

# Read from maiml
for i in range(fileNum):
    maiml = maimlfiles[index]
    index += 1
    maimltree = ET.parse(maiml)
    root = maimltree.getroot()

    # to row data
    for e in root.iter('{http://www.maiml.org/schemas}property'):
        key = e.attrib['key']
        description = ""
        for child in e.iter('{http://www.maiml.org/schemas}description'):
            description = child.text
        row.append([key, description])

# Write to CSV
df = pd.DataFrame(row, columns=HEADERS)
df.to_csv ('path/to/file.csv')
```

Original MaiML File

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

```
<maiml version="1.0" features="nested-attributes"
        xmlns="http://www.maiml.org/schemas"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="rootObjectType">
  <document id="MaimlSampleDoc">
  </document>
  <protocol id="MaimlSampleProtocol">
    <uuid>5ac5ab10-866d-4ea2-b9f0-f9394d66b66c</uuid>
    <name>SEM_protpcol</name>
    <method id="methodID">
      <program id="programID">
        <materialTemplate id="MM_T">
          <uuid>33e3c15c-81fb-4275-95f2-16504d0d16ce</uuid>
          <name>material_XdashTemplate</name>
          <property xsi:type="stringType" key="exm:MaterialLotNo">
             <description>Lot No. of Material</description>
            <value>Lot No.</value>
            <property xsi:type="stringType" key="exm:LotNumber"></property>
          </property>
          <property xsi:type="stringType" key="exm:MaterialType">
            <description>Type of Material</description>
            <value> Bulk </value>
            <property xsi:type="doubleType" key="exm:Thickness" formatString="0.00"
                      scaleFactor="1.00" unit="mm">
              <value> 10.00 </value>
            </property>
          </property>
          <property xsi:type="stringType" key="exm:Comment">
            <description>Comment for Material</description>
            <value> "Color is black." </value>
          </property>
          <placeRef id="pr1" ref="MM-1"></placeRef>
          <placeRef id="pr2" ref="MM-2"></placeRef>
        </materialTemplate>
        <conditionTemplate id="CC-1T">
          <uuid>faa939ab-e634-4e33-99d0-bfd1545f4317</uuid>
          <name>SEMconditionTemplate</name>
          <description>SEM Condition</description>
          <property xsi:type="doubleType" key="exm:AcceleratingVoltage" formatString="1.00"
                     units="kV" sacleFactor="1.0">
            <value> 5.00 </value>
          </property>
          <property xsi:type="stringType" key="exm:Note">
            <value> Low dose condition for redusing electron damage </value>
          </property>
          <placeRef id="pr4" ref="CC-1" ></placeRef>
        </conditionTemplate>
        <resultTemplate id="RR-2T">
          <uuid>cc77bc35-267b-4d25-a76c-6b726a849028</uuid>
          <name>SEMImageTemplate</name>
          <property xsi:type="uriType" key="exm:SEMImageUri">
          </property>
          <property xsi:type="uriType" key="exm:SEMOutConditionUri">
          </property>
          <placeRef id="pr6" ref="RR-2" ></placeRef>
        </resultTemplate>
      </program>
    </method>
  </protocol>
</maiml>
```

Output CSV File

```
,key,description
0,exm:MaterialLotNo,Lot No. of Material
1,exm:LotNumber,
2,exm:MaterialType,Type of Material
3,exm:Thickness,
4,exm:Comment,Comment for Material
5,exm:AcceleratingVoltage,
6,exm:Note,
7,exm:SEMImageUri,
8,exm:SEMOutConditionUri,
```

11. Frequently Asked Questions (FAQ)

11.1 Use Cases

11.1.1 Indicating Experiment Data Flows

11.1.1.1 What are the differences between <material>, <condition>, and <result> elements?

<material> elements indicate what is being measured/analyzed. <condition> elements indicate parameters for measurements/analysis. <result> elements express the data obtained using measurement/analysis operations. All three types can be used as input or output values.

For example, pretreatment processes for measurement/analysis will output <material> elements for the next measurement/analysis. If a <result> element output from a measurement instrument is used for the next analysis, then the <result> element can be used directly as an input value without making it a <material> element.

Reference: <result> elements indicated in **JIS K 0200 6.4.3**

11.1.1.2 How should experiment protocols be indicated?

<pnml>, <instruction>, <materialTemplate>, <condtionTemplate>, and <result> elements can also be used to express experiment protocols.

Example of using raw material A is heated in an operation according to condition B to obtain material C

```
<method>
  <uuid></uuid>
  <pnml>
    <!-- Because there are three elements for raw material A, condition B, and material C,
          three <place> elements are prepared and id elements named. -->
    <place id="MM1"></place>
    <place id="CC1"></place>
    <place id="MM2"></place>
    <!-- Because there is one heating operation,
    one <transition> element is prepared and id element named. -->
    <place id="PP1"></place>
    <!—The protocol process flow is indicated using <arc>elements. -->
    <arc id="MM1PP1" source="MM1" target="PP1">
    <arc id="CC1PP1" source="CC1" target="PP1">
    <arc id="PP1MM2" source="PP1" target="MM2">
  </pnml>
  <program>
      <uuid></uuid>
      <instruction id="firstHeating">
        <uuid></uuid>
        <transitionRef ref="PP1"></transitionRef>
      </instruction>
      <materialTemplate id="rawMaterial">
        <uuid></uuid>
        <placeRef ref="MM1">
      </material>
      <materialTemplate id="heatedMaterial">
        <uuid></uuid>
        <placeRef ref="MM2">
      </material>
      <conditionTemplate id="heatingCondition">
        <uuid></uuid>
        <placeRef ref="CC1"></placeRef>
    </condition>
  </prograrm>
</method>
```

11.2 Elements Used to Indicate Structure

11.2.1 Elements Related to Global Elements

11.2.1.1 How should UUID values be generated for global elements?

There are two types of methods for generating UUID values used for global elements. One is to generate the UUID values from <name> elements that can also be used as a unique global element based on Version 3. The other is to generate UUID values from random numbers based on Version 4.

For the first method, because the <name> elements include a QName type value, namespaces are used to append names with a modifier (XML modified names) to provide uniqueness. Then that character string is expected to be used to generate unique UUID values. Identical UUID values are generated by the identical character strings. Next, the method of generating values using the Python language is described below.

```
import uuid
print(uuid.uuid3(uuid.NAMESPACE_URL, "http://www.maiml.org/ontology/Microscopy:Magnification"))
```

The second method generates values using the Python language, as indicated below. Because it involves random numbers, the number changes each time the process is executed.

```
import uuid
print(uuid.uuid4())
```

11.2.1.2 Are global elements ever assigned the same UUID value?

If Version 3 or 5 is used for UUID values, namespaces are used to express <name> elements with an XML modified name to ensure uniqueness before conversion to UUID values. Using namespaces, uniqueness is ensured by vendors, users, or others.

In contrast, if Version 4 is used, UUIDvalues are generated using random numbers. Because 122-bit random numbers are used, the probability of generating an existing value is one in 2.30 quintillion (1/2122/2 = 1/2.3x1018), which can normally be ignored. Furthermore, even if an existing value is generated, other element names or element values, such as other hash values, global element names, or id values, can be used for verification, so the risk of it actually becoming a problem is even less.

11.2.1.3 What are the differences between <name>, <description>, and <annotation> elements used as child elements of global elements?

<name> elements ensure uniqueness using namespaces. However, beware that “name” is used differently than in natural language.

<description> elements are used to describe the meaning of global elements. They are expected to be used as terms in headers, such as for spreadsheet, visualization, or other software.

<annotation> elements are expected to be used to indicate detailed descriptive or other information about global elements.

11.2.2 Elements Used to Indicate Element Attributes

11.2.2.1 How should id attributes be specified?

id attributes use ref attributes or various elements to reference other elements within files. Also refer to the case described in **clause 9**.

11.2.2.2 How should ref attributes be assigned?

ref attributes are used to reference id attributes within files. Also refer to the case described in **clause 9**.

11.2.3 Elements Used to Indicate Relationships between Elements

11.2.3.1 Is there a way to omit indicating <material> elements with almost identical descendant elements?

Once an <instanceRef> element has been used to reference the material element, then only the descendant elements that include changes can be rewritten.

Also, for prespecified elements and values, template id attributes indicated using a <materialTemplate> element can be referenced using ref attributes.

11.3 Elements Related to How General Purpose Data Containers are Used

11.3.1 Elements Related to Key Attributes

11.3.1.1 How should the uniqueness of key attributes in general purpose data containers be ensured?

key attributes are a QName type elements used as keywords with a modifier specified in a namespace. They are expected to be used to ensure uniqueness. Therefore, converters and other means of creating MaiML files are premised on the uniqueness of specified keywords being ensured within namespaces.

11.3.2 Elements Used to Indicate <value> Element Content

11.3.2.1 When a program loads a space-delimited list of content for <value> elements, at least one blank character (space, tab, carriage return, or line feed) from the beginning and end of lists and at least one blank character from between respective listed items are directly included. How should they be handled?

XML Schema Part 2: **2.5.1.2** List Data Types in the section about data types, specifies that blank characters should be collapsed.

However, unless the XML schema file is loaded by the program, XML parser library used to process XML code cannot interpret content in listed form.

Even when using an XML parser that supports verification using list-based schemas, blank characters can sometime fail to collapse or light weight and fast XML parsers might not be supported by the schema.

In such cases, space-delimited lists in the content of <value> elements must be processed according to list type specifications, as indicated below.

- “Blank” characters refers to space (U+0020), tab (U+0009), line feed (U+000A), and carriage return (U+000D) characters.
- When collapsed, black characters are first replaced. Specifically, tab (U+0009), line feed (U+000A), and

carriage return (U+000D) characters are replaced with space (U+0020) characters.

- Next, all the spaces before and after the content are deleted and each string of one or more spaces located between listed items is replaced with one space.
- Then the content is divided into individual list items that are converted to a form for the programming language.

Based on the above, the person that created the converter then considers the following factors.

- If at all possible, avoid inserting a line feed or indent for organizing XML code immediately after a <value> element start tag, immediately before a <value> element end tag, or between items in the list.
- Use <value> elements to divide lists, rather than inserting a line feed or indent for organizing XML code.

11.3.2.2 Using the program to load a <value> element that contains an extremely large space-delimited list can result in a memory error or slow down interpretation of data.

XML specifications do not limit the amount of text that can be included as content in elements.

However, program operating environments usually limit the file size or amount of text mainly due to security requirements.

- If the text limits of the XML parser used by the program can be changed, then try the methods specified in the XML parser manual. However, it is necessary to be aware that methods that permanently increase the quantity limits for other programs that use the same XML parser may become a target for denial of service (DoS) attacks.
- Next, consider processing the XML file as text to divide lists for parallel processing. Also beware that if a combination of programs is used to split character strings or convert data types, they might have different processing speeds depending on the selected method.

Based on the above, the person that created the converter then considers the following factors.

- Divide <value> elements to prevent the byte size of content in <value> elements from becoming too large.
- xs:double type list items with maximum precision, including the exponent, consist of about 20-byte character strings. In that case, generally about 50,000 data points can be stored in each <value> element per 1 M bytes of temporary memory available for use by the XML parser. However, preferably the size of split lists should be decided based on not only limit values for the XML parser used by the converter, but also factors such as the XML parser used by the user, memory efficiency, and processing speed.

11.3.2.3 Character strings such as “&lt;,” “&#x20;,” “&#32;,” and “&Binning;” are included in <value> element content.

Such character strings are referred to as entity references used as a substitute for actual characters or character strings in names or numbers. Entity references must be replaced with the defined character or character string and interpreted.

- Entity references must be converted to the correct character string using the entity reference and character string conversion functionality available in the XML parser used by the program.
- Some XML parsers configured to default settings do not analyze “&Binning;” or other DTD entities, so check the corresponding XML parser manual.

If XML parser functionality cannot be used, use the following procedure.

- “&lt;,” “&gt;,” “&quot;,” “&apos;,” and “&amp;” entities are referred to as predefined entities used to express the four XML marking characters “<,” “>, “",” and “',” and also the “&” character used for entity references. A character string replacement function is used to restore the actual characters to which the entities refer.

- "&#x20;" and "&#32;" entities are referred to as XML numbered entities and are replaced with corresponding base-16 and base-10 Unicode characters.
- "&Binning;" and other entity references that are not predefined or numbered are replaced with character strings defined in the DTD document type (DOCTYPE) declaration area indicated before <maiml> elements.

11.3.2.4 Character strings such as "\u0020," "\t," "\r," "\n," or "\\" are included in <value> element content.

These character strings are referred to as printf escape sequences for the C language and must be replaced with space, tab, line feed, carriage return, or reverse solidus (backslash) characters, respectively.

Other methods are also available as escape sequences, so check whether a definition that uses a <property> element exists and, if not, check with the person that created the converter.

The converter creator should be careful of the following.

- Reverse solidus (backslash) characters were being replaced with a different character when using local language-specific ASCII character sets, as indicated in **Appendix A**. For example, for the Japanese language, they were replaced with a Japanese yen symbol.
- Such characters are assigned a different character code in Unicode, but due to incompatibility with ASCII code some program languages or editors are not able to distinguish it as a backlash. Therefore, also consider replacing such characters with hexadecimal (base-16) escape sequences (\u00A5 in the case of the yen symbol).

11.4 Elements Used for Confidentiality

11.4.1 Is the a convenient way to check whether confidentiality was applied?

In cases such as the following, the confidentiality status can be checked by replacement with the <EncryptedData> element.

```
<EncryptedData>
<!-- Omitted -->
</EncryptedData>
```

11.4.2 Shared keys for applicable key methods or secret keys for asymmetric key methods are necessary for canceling confidentiality, but if a shared or secret key is permanently provided to an outside organization to permit access to confidential information, it could cause information leakage due to inappropriate access to the key. Are there any appropriate countermeasures?

(1) MaiML files and keys can be controlled using a cloud system.

(2) Entire original MaiML files without confidentiality can be encrypted and managed in an area only accessible to the owner organization

(3) Each time confidential data is provided to a user of confidential data, confidentiality can be applied to the original parent MaiML file by the Elliptic-curve Diffie-Hellmankey exchange method.

(4) The Elliptic-curve Diffie-Hellman key exchange method requires both the confidential data provider and user to use both a secret key and public key. For example, leakage of confidential information in data can be prevented in three ways. (1) Provide different keys to each data recipient. (2) Because a key and MaiML file with confidentiality canceled cannot be downloaded directly, the confidentiality is applied and

canceled by a cloud system, with keys provided to data users on a subscription basis for only viewing the confidential portions of data or obtaining it from an API. (3) Both methods (1) and (2) are used.

(5) For MaiML data files that can be released arbitrarily, security risks can be presumably minimized by encrypting them with the strongest encryption method available before they are released and then not permitting access to the keys necessary for removing confidentiality.

11.4.3 Are there any recommended precautions for managing theelectronic keys or digital certificates used for XML signatures and encryption?

(1) Avoid using electronic keys or digital certificates for signatures in organizational management, commercial transactions, or sales/distribution software. A leaked key/certificate could be misused by a malicious attacker or cause a breakdown of organization management, loss of trust, or other problems.

(2) Creating an electronic key or digital certificate for each member of an organization and including personal information could cause personal information to leak outside the organization from XML signatures.

(3) Systems must be operated so that the number of organization members permitted to use electronic keys or digital certificates is kept to a minimum.

(4) Electronic keys and digital certificates used for XML signatures should be created by entire organizations or by respective subordinate organizations and installed in devices used to apply or verify XML signatures or to encrypt or decrypt XML. Then the electronic keys and digital certificates should be carefully managed to prevent access except by authorized programs or organization members.

(5) Electronic keys and digital certificates used for XML confidentiality should be created separately for each type of data and each data recipient, but due to the difficulty of managing the keys, they should preferably be managed in a cloud system used to manage MaiML files and keys.

(6) The original electronic keys and digital certificates at the data provider and backup copies and so on located in the cloud system should be stored on recording media kept in a fire-resistant safe, for example, and never left on a shared server or external device.

(7) Confidential MaiML files and keys stored in the cloud or other system should be regularly rotated, and have expiration dates assigned and processed.

(8) In particular, due to the risk of former employees at data providers becoming attackers, a key rotation and expiration system should be executed whenever a manager with key access leaves the organization.

11.4.4 If data tampering occurs, is there a risk the confidential portions being disclosed?

Even if tampering occurs, is there a risk of confidential portions being disclosed?

11.4.5 What methods are available to prevent tampering methods such as attackers deleting the XML signature in a MaiML file and then signing the file again after tampering with the data or attackers falsifying the signer name (typically the owner organization), changing the creation date, or signature date?

(1) Use the <chain> elements described in **JIS K 0200 B.10.1**. If a MaiML file with tampering is related to multiple MaiML files with no tampering, then they can be used to detect tampering. However, if an entire related file was tampered with or tampering with the signer name, creation date, or signature date was detected, other methods must also be used.

(2) Beware of the possibility that the owner information was submitted by the MaiML file creator (not necessarily the owner) or attacker. To prevent that possibility, use digital certificates issued by a third-party certification agent for XML signatures. By using time stamp signatures in combination with digital certificate expiration dates and validity period information, third-party certification agents and time stamp certification agents will guarantee that a file was signed by a signer (typically the owner organization) who was authorized to sign the file on the signature date. In addition, by comparing the signature date and creation date, the time stamp certification agent can guarantee that the signature date is after the creation date.

(3) Instead of method (2), equivalent functionality as provided by (2) can be provided by establishing a cloud system for managing the MaiML files and electronic keys before and after signatures are applied. That requires the cloud system operator to guarantee that the correct signer (typically the owner organization) signed the file on a signature date after the creation date.

11.5 Validity of Data Format

11.5.1 Is there a way to confirm whether the created MaiML file conforms to the proper format?

A grammar guidance called an XML Schema is now available for MaiML files. The proper format can be confirmed using that document. Used in combination with LINQPad software, the format can be checked in detail.

In addition, minimal checks can also be performed using the Python language. An example is described below.

```
from lxml import etree

XMLFILE_PATH = "/.../aaa.xml"
XSDFILE_PATH = "bbb.xsd" // Checks whether there is correspondence to multiple files.

def validate() -> bool:
    xml_doc = etree.parse(XMLFILE_PATH)
    xsd_doc = etree.parse(XSDFILE_PATH)

    xmlschema = etree.XMLSchema(xsd_doc)
    result = xmlschema.validate(xml_doc)

    return result
```

11.6 Other FAQ

11.6.1 If multiple types of measurement data were acquired from the same sample, should the corresponding MaiML files be combined into one file?

If multiple types of measurement data were acquired from the same sample, there is no need to consolidate MaiML files into one file. The fact that the <uuid> elements in the <material> elements contain the same value ensures that the data is from the same sample. Also refer to the use case about using data, described in **10.2** of this guideline.

Appendix A Guideline for Characters Used in MaiML

A.1 Character Set and Encoding Method Used in MaiML

Normally, the character set used for MaiML files is Unicode and the encoding method is UTF-8 (without BOM).

In addition to Unicode , various other ASCII character sets or extensions thereof, continue to be used in a wide variety of programs since before Unicode was established.

Note that if text included in measurement/analysis data originally stored using a different character set and encoding method, then the data must be converted to Unicode before the data can be stored in a MaiML file.

Note 1 A character set (a collection of encoded characters) is a set of characters with a unique number (character code) assigned to each character. Unicode is the character set recommended for XML. Examples of non-Unicode character sets include ASCII and its extensions specified by ISO/IEC 646 and ISO/IEC 8859 and the ASCII extension for Japanese language specified by **JIS X 0201**, and the JIS character sets for Japanese kanji characters (specified by **JIS X 0208**, **JIS X 0212**, and **JIS X 0213**.)

Note 2 Encoding (character coding system) refers to converting character codes into byte sequences (encoding) by bit operation, table conversion, or other method. UTF-8 is the encoding method recommended for XML. Encoding methods other than UTF-8 include UTF-16BE, UTF-16LE, UTF-32BE, and UTF-32LE methods for the same Unicode character set as UTF-8 and the JIS (ISO/IEC2022-JP), SHIFT-JIS, and EUC-JP methods for encoding the ASCII Japanese language extension character set.

Note 3 If Japanese characters are used in a MaiML file based on Unicode characters, file users in non-Japanese operating environments must set up extended operating system functionality used in the file in order to use the additional fonts used in the MaiML file. Similar measures are also required if other languages are used. Some languages include language-specific characters, which are difficult to input without a language-specific keyboard and character input system.

Note 4 If the ASCII extended character set and encoding methods for Japanese language are used, MaiML file users working in non-Japanese language environments will have to use a software module to provide compatibility with the ASCII extended character set and encoding methods for the Japanese language and then expressly switch between the respective character sets and encoding methods. Similar measures are also required if other languages are used. Using non-Unicode character sets is not recommended.

A.2 XML Declaration and Encoding Method

If the recommended character set and encoding method are used, a "1.0” version number is indicated in an XML declaration at the top of the MaiML file to indicate conformance with XML 1.0 and “UTF-8” (or lowercase “utf-8”) is indicated as the encoding method. Then select the encoding method to UTF-8 (without BOM) when saving the MaiML file.

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

Note 1 Though a recommendation to use XML 1.1 has already been issued, due to the limited adoption, the program XML processing system might not read the XML code properly if “1.1” is indicated as the version.

Note 2 If the file encoding system does not match the XML declaration, then the program’s XML processing system might not be able to read the file properly.

XML Declaration	File Encoding Method	Schema Checker
version="1.1"	—	Stopped during loading of XML Declaration
encoding="UTF-8"	“UTF-16BE” or “UTF-16LE”	Stopped while reading the first line.
encoding="UTF-16BE” or encoding="UTF-16LE"	UTF-8 (without BOM)	
encoding="UTF-8"	SHIFT-JIS or EUC-JP	Stopped while reading line with Japanese characters. Beware that no error occurs if there is no Japanese character data.
encoding="SHIFT-JIS” or encoding="EUC-JP"	UTF-8 (without BOM)	
encoding="UTF-8"	UTF-8 (with BOM)	No error occurs.

Note 3 Encoded as 2 bytes by UTF-16BE and UTF-16LE Unicode character sets, but as 1 to 4 bytes by UTF-8, which triggered an error at the first line.

Note 4 With the SHIFT-JIS, EUC-JP, and UTF-8 character sets, which are ASCII character set extensions and encoding methods for Japanese language, the control code 0 and basic Latin characters are encoded as the same 1 byte, but when any other character is encoded, the byte sequence deviates from patterns possible for UTF-8 byte sequences and results in an error at a line that includes a Japanese character.

Note 5 If a file is saved using UTF-8 (with BOM) encoding, an XML processing system exists that generates an error when the 3 bytes “EF BB BF” encoded by UTF-8 from U+FEFF, which is the byte order mark (BOM), are loaded when saving the file. Note that the XML processing system used by the schema checker does not generate an error because it deletes the BOM from the beginning, but does generate an error in some cases if problem settings are changed.

A.3 Overview of Unicode Character Regions and Precautions

Unicode classification names are assigned based on ranges of character codes. An overview and precautions are indicated below.

Character Code Range	Class Name	Remarks
U+0000 to U+001F and U+007F	Control code 0 (C0)	This group of control code characters are used for communication, window control, editing, printing, etc. Not used for MaiML files, except for horizontal tab (U+0009), LF (U+000A), and CR (U+000D).
U+0020 to U+007E	Basic Latin	Group of characters for language-neutral (US) space (U+0020), upper/lower alphabetic and numeric characters, and symbols. All but a few can be used in MaiML files without problems. Fully matches character ranges specified in US-ASCII (ISO/IEC 646-US) and **ISO/IEC 8859**-1, which are language-neutral (US) ASCII character sets. Beware that it does not include some characters for specific languages that replace standard ASCII characters, such as “¥” and “‾” characters in **ISO-646-JA** and **JIS X 0201** and the “£” and “‾” characters in **ISO-646-GB**.
U+0080 to U+009F	Control code 1 (C1)	Group of control code characters based on the first 8 bits of the control code 0 (C0). Not used for MaiML files
U+00A0 to U+00FF	Latin-1 Supplement	Includes additional language-neutral (US) Latin characters and symbols. Requires caution for use in MaiML files. Fully matches same character range as specified in **ISO/IEC**

		8859-1, which is a language-neutral (US) ASCII character set. ASCII character set with various characters substituted with characters for specific countries, such as “¥,” “£,” and “¯” assigned to U+00A5, U+00A3, and U+00AF, respectively, for example.
U+0100 to U+10FFFF	(Other)	Includes additional characters, symbols, etc., such as characters from the code pages of ASCII and its extended character set, characters for archaic and other languages. Requires caution for use in MaiML files. Reassigned character codes, including JIS kanji characters (**JIS X 0208, JIS X 0212, and JIS X 0213**).

The use of characters from the Latin-1 supplement and other character ranges is significantly affected by keyboard layout, character input systems, and fonts. In particular, if used for <name> elements or key, id, or ref attributes, the characters could hinder entering values in search algorithms for AI, DX, and big data analysis applications. Avoiding their use should be considered.

In addition, Unicode has added many single-byte symbols, such as for units and Greek symbols, that are based on JIS double-byte characters. The method used to add characters varies depending on the character, such as adding both single and double-byte versions and then also dividing the characters by shape or by adding only a single-byte version and switching between single-/double-byte versions based on the font. If both are added, then using the single-byte version is recommended. Careful consideration is required when using in MaiML files.

Note 1 A “font problem” refers to the following.

- Some fonts and programs display shapes based on characters substituted for characters in ASCII or extended ASCII sets, rather than based on the shape of the character normally assigned to that character code.
- The character resembles characters for multiple character codes depending on the font. Unicode characters are named, so that both the shape and name must be checked in the character code table and entered, rather than simply accepting keyboard symbols or conversion candidates indicated by the character input system.
- Character shapes are not available for character codes, with the shape of substitute characters displayed instead. That can make it difficult to distinguish between shapes.
- Some languages use a combination of zero-width characters that are not displayed. If zero-width characters are included in a character string, it can be difficult to visually determine whether a character is shown.
- Character strings cannot be compared/searched properly if characters visually appear the same but have different character codes.

Note 2 Character codes can also cause problems when entering characters via a keyboard or character input system. Specifically, except when a native local language user uses a keyboard with the local language layout or a character input system with local fonts, character codes can cause problems determining whether the correct character codes were entered.

Note 3 When the recommended UTF-8 encoding method is used for MaiML files, the length of byte strings can vary for each character, resulting in complex bit operations being used to calculate byte values and in character codes that do not match byte strings, except for basic Latin characters and the control code 0. Checking character codes requires conversion to UTF-

16LE, UTF-16BE, or other encoding that outputs most character codes directly as byte strings, and then opening the results in a binary editor, and then checking the character codes.

The following are some examples of characters that require particular caution.

Characters that Require Caution	Remarks
Basic Latin Characters Reverse solidus “\” (U+005C)	US-ASCII (ISO/IEC 646-US) characters that are substituted with other characters in country-specific ASCII character sets. This is used as an escape sequence in typical programs and could become mixed into MaiML files. For some languages or fonts, the character code is directly interchangeable with “¥” or other symbols in **ISO-646-JA** or **JIS X 0201**, which can result in displaying the shapes of substitute characters. Therefore, language information should be provided.
Latin Supplement-1 Yen symbol “¥” (U+00A5)	In Japanese ASCII character sets, the US-ASCII (ISO/IEC 646-US) character is substituted with a “¥” symbol. For Japanese fonts, the same character code is used to display the substituted character shape in some cases. Therefore, language information should be provided. Caution is required because it can be difficult to distinguish between “\” and “¥” characters when using certain combinations of keyboards, fonts, or software.
Latin Supplement-1 Degree symbol “°” (U+00B0)	Symbol shared by SI that indicates degree It can cause problems due to the following similarly shaped characters. • Masculine ordinal indicator “º” (U+00BA) • Single-byte katakana “ﾟ ” (U+FF9F) symbol for semi-voiced sound Japanese fonts can sometimes display it as a double-byte “°” character, but the character code is identical. Japanese fonts on older processing systems sometimes display it as a single-byte horizontal bar “ｰ” for indicating long sounds in katakana, which used to be displayed by **JIS X 0201**. Latin alphabet-based keyboards can be used to enter the “º” mark attached to ordinal numbers in the Latin language. That requires caution to prevent confusion. It requires caution even if using a character code chart to enter the character, because it is easily confused with the adjacent “º” code. The SI unit for Celsius requires using “℃” (U+2103), rather than using this character in combination with an uppercase C.
Latin Supplement-1 Middle dot “·” (U+00B7)	This character can be used in XML name types (xs:Name) and its derivative types. It can cause problems due to the following similarly shaped characters. • Greek ano teleia (semicolon) “·” (U+0387) • Bullet “•” (U+2022) • Double-byte middle dot “・” (U+30FB) • Single-byte middle dot “･” (U+FF65) They have different font shape, but it is difficult to distinguish without side-by-side comparison.
Latin Supplement-1 Latin capital A with ring above “Å” (U+00C5)	Non-SI unit no longer used for SI, but used to be used as symbol for units of atomic size. Not recommended for use. It can cause problems due to the following similarly shaped characters. • JIS symbol for Angstrom “Å” (U+212B) “Å” originated as a JIS symbol but is indicated as a double-byte character in Japanese fonts. It appears as a single-byte “Å” in European fonts, which is indistinguishable from U+00C5. The “Å” (U+00C5) character is recommended for the Angstrom symbol. The double-byte Å character (U+212B) is not recommended.
Greek and Coptic A “Α” (U+0391)	Greek letter easily confused with language-neutral (US English) character.

	It can cause problems due to the following similarly shaped characters. • Basic Latin character “A” (U+0041) In European fonts shaped the same as basic Latin character “A” (U+0041). In Japanese fonts it is indicated as a double-byte “A” character, but the character code is identical. All Greek letters have different character codes than language-neutral (US English) letters, even if identically shaped. On Greek keyboards, beware that all language-neutral (US English) letters are substituted with Greek letters and character codes for Greek letters are entered. For the Japanese language, caution is required when the character conversion key in combination with other keys. The characters A and B are easily distinguish in a Japanese font, but are indistinguishable even when mixed with European fonts. • “ABC” entered using non-Greek keys:“&#x0041;&#x0042;&#x0043;” • “ΑΒΓ” entered using Greek keys: “&#x0391;&#x392;&#x0393;” • “ΑΒΓ” converted to Japanese character conversion: &#x0391;&#x0392;&#x0393;” • “ABΓ” entered and converted with Japanese keys: “&# x0041;&# x0042;&# x0393;”
Typical punctuation Zero width non-joiner (U+200C)	Zero-width character sometimes inserted between prefix (ex: Auf-) and root of compound words, such as in German. German keyboards include a symbol for entering zero width non-joiner characters, the character cannot be entered with many other keyboard layouts. Caution is required because it is difficult to determine whether the non-displayed character is inserted. • "Auflage” without ZWNJ: “&#x0041;&#x0075;&#x0066;” “&#x006C;&#x0061;&#x 0072;&#x0067;&#x0065;” • "Auflage” with ZWNJ: “&#x0041;&#x0075;&#x0066;&#x200C;” “&#x006C;&#x0061;&#x 0072;&#x0067;&#x0065;” (By moving the cursor, the zero width character can barely be determined.)

Note 4 In XML, “#x0041” is referred to as a numbered entity for Unicode character “A” (U+0041) and “&#x0041;” and a numbered entity reference. In XML files, all numbered entity reference Unicode characters are treated directly as Unicode characters.

Appendix B Overview of XML Modified Names and Naming Guideline

Programs treat all XML modified names as an XML name type (xs:Name) or a derivative thereof. They are not names as used in natural language. Therefore, similarly to other programming languages, names must conform to naming rules. The following is an overview of XML modified names used in MaiML files and provides a guideline for naming them.

B.1 XML Name Type (xs:Name)

Rules for naming XML name types (xs:Name) used to name XML modified names are defined in section 2.3 Common Syntactic Constructs of Extensible Markup Language (XML) 1.0 (Fifth Edition), as follows.

```
NameStartChar    ::=     ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] | [#xF8-#x2FF] |
[#x370-#x37D] | [#x37F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] |
[#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] |
[#x10000-#xEFFFF]
NameChar         ::=     NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] | [#x203F-
#x2040]
Name             ::=     NameStartChar (NameChar)*
```

Note 1 In this case, [A-Z] and [a-z] indicate 52 language-neutral (US English) letters of the alphabet. [0-9] indicate ten language-neutral (US English) numbers. [#xC0-#xD6] indicates the range of Unicode characters from “À” (U+00C0) to “Ö” (U+00D6).

Note 2 NameStartChar indicates the naming rule for the first character in XML names (Name) and NameChar the rule for the second and other characters thereafter.

Considering the international distribution of MaiML files, it is advisable to limit the characters used in XML name types (xs:Name) to only basic Latin Unicodecharacters, as described above, and especially avoid using non-displaying zero-width characters.

As a reference, a list of the basic Latin Unicode characters that satisfy the requirements defined in section 2.3 Common Syntactic Constructs of the Extensible Markup Language (XML) 1.0 (Fifth Edition) is indicated below.

```
NameStartChar    ::=     ":" | [A-Z] | "_" | [a-z]
NameChar         ::=     NameStartChar | "-" | "." | [0-9]
Name     ::=     NameStartChar (NameChar)*
```

B.2 No-Colon Name Type (xs:NCName)

The no-colon name type (xs:NCName) is an XML modified name without a colon that serves as the basis for prefixed XML modified names (xs:QName) used for <name> elements and key attributes, for (xs:ID) ID types used for id attributes, and for (xs:IDREF) ID reference types used for ref attributes.

As defined in “Namespaces in XML 1.0 (Third Edition), 3 Declaring Namespaces,” and indicated below, the no-colon name type is an XML name type (xs:Name) with, as the name suggests, the colon “:” (U+003A) removed.

```
NCName::=        Name - (Char* ':' Char*)      /* An XML Name，minus the ":" */
```

As a reference, a list of basic Latin Unicode characters that satisfy the definition for the no-colon name type is indicated below.

```
NCNameStartChar::=       [A-Z] | "_" | [a-z]
NCNameChar       ::=     NCNameStartChar | "-" | "." | [0-9]
NCName::=        NCNameStartChar (NCNameChar)*
```

The following applies when limiting the characters available for no-colon name types (xs:NCName) to only basic Latin Unicode characters.

- The only characters that can be used at the beginning of the name type are upper or lowercase language-neutral (US English) alphabetic or low line “_” (U+005F) characters.
- The only characters that can be used for the second and other remaining characters are the characters that can be used at the beginning, language-neutral (US English) numbers, hyphen-minuses “-” (U+002D), or full stops “.” (U+002E).

The following names are valid no-colon name types (xs:NCName) limited to basic Latin characters.

- GasChromatographyMassSpectroscopy: Indicated by method referred to as PascalCase.
- Gas_Chromatography_Mass_Spectroscopy: Indicated by method referred to as Title_Case.
- GC-MS: Indicated by method referred to as kebab-case.
- material.type: Indicated by method referred to as dot.separated.case.
- PID001234
- _3d_Parameters: Indicated by method referred to as Title_Case. Because names cannot start with a number, a low line “_” (U+005F) is used.

The following names are invalid no-colon name types (xs:NCName) limited to basic Latin characters.

- Gas Chromatography Mass Spectroscopy: Blank characters cannot be used.
- Gas&#x20;Chromatography&#x20;Mass&#x20;Spectroscopy: Blank characters cannot be used even if a numbered entity reference is used.
- GC/MS: Forward slashes “/” cannot be used as delimiter characters.
- material，type: Commas “，” cannot be used as delimiter characters.
- 001234: Names cannot start with a number.
- .3d.Parameters: The only symbol that can be used at the beginning is a low line “_” (U+005F) character.

Examples of XML modified name naming rules are indicated below.

Naming Rule Name	Naming Rule
PascalCase	Only the first character of individual words, initials or abbreviations is capitalized, with delimiter characters deleted or consolidated.
camelCase	Only the first character of individual words, initials or abbreviations is capitalized, delimiter characters must be deleted or consolidated, and the first character must be lowercase.
UPPERCASE	All individual words, initials or abbreviations must be uppercases, with delimiter characters deleted or consolidated.
lowercase	All individual words, initials or abbreviations must be must be lowercase, with delimiter characters deleted or consolidated.
snake_case or underscore_case	All individual words, initials or abbreviations must be must be lowercase, with delimiter characters replaced with low line “_” characters and consolidated. However, this naming rule name is sometimes used without regard for upper/lowercase.
Title_Case or Pascal_Snake_Case	Only the first character of individual words, initials or abbreviations is capitalized, with delimiter characters replaced with low line “_” characters and consolidated.
MACRO_CASE or CONSTANT_CASE	All individual words, initials or abbreviations must be uppercases, with delimiter characters replaced with low line “_” characters and consolidated.
dot.separated.case	All individual words, initials or abbreviations must be must be lowercase, with delimiter characters replaced with a full stop “.” and consolidated.

	However, this naming rule name is sometimes used without regard for upper/lowercase.
kebab-case	All individual words, initials or abbreviations must be must be lowercase, with delimiter characters replaced with hyphen-minus “-” characters and consolidated. However, this naming rule name is sometimes used without regard for upper/lowercase.

Note 1 Naming rule names with naming rules applied to the displayed naming rule are used. For example, “PascalCase” is used instead of “pascal case.”

Note 2 Prerequisites for applying these rules include deleting or replacing invalid symbols and other characters in advance.

Note 3 In terms of applying the above rule for indicating individual words with upper or lowercase characters, diversity can often be achieved by applying the rule to compound words such as PascalCase or camelCase, permitting forms such as PascalCase_With_Underscore or camelCase_with_underscore, or permitting the initials or abbreviations to be in all uppercase.

Note 4 The naming rules can be referred to by other names than indicated here.

URL values specified by a prefixed name type (xs:QName), described below, can be used as a terminology resource URL defined by an RDF/OWL, for AI data analysis, or other uses. In that case, the no-colon name type (xs:NCName) that is part of that will serve as an identifier for the class or property defined by RDF/OWL. Though the data structure of the original measurement/analysis data can be used to name the identifier, the structures are not necessarily named optimally, so it may be appropriate to name them based on English expressed used to describe the data structures with natural language.

On the other hand, if naming rules for no-colon name types (xs:NCName) are restricted to Unicode basic Latin characters and also hyphen-minus “-” (U+002D) and full stop “.” (U+002E) characters are excluded, then the naming rules for the local name (LocalPart) will provide greater compatibility with naming rules for many programming languages and databases.

If optimal naming is not a concern, then the no-colon name type (xs:NCName) can be named directly based on the class name and class attribute name of the data or the table name and column name of the database.

For programming languages with a scope-resolving operator such as namespaces or member-selected operators, dot.separated.case can be used with full stops “.” (U+002E) as delimiter characters.

B.3 Prefixed Name Types and XML Modified Names (xs:QName)

Prefixed name types (xs:QName) are XML modified names used in <name> elements and key attributes.

They are defined in “Namespaces in XML 1.0 (Third Edition), 3 Declaring Namespaces,” as indicated below.

```
QName   ::=         PrefixedName | UnprefixedName
PrefixedName      ::=         Prefix ':' LocalPart
UnprefixedName    ::=         LocalPart
Prefix     ::=      NCName
LocalPart         ::=         NCName
```

Prefixed name types (xs:QName) are formed by combining a prefix and a local name (LocalPart) with a colon “:” (U+003A) in between or formed from only a local name, where the prefix (Prefix) and local name (LocalPart) satisfy the naming rules for no-colon name types (xs:NCName).

The prefix only needs to be different than other prefixes in the same MaiML file, but the URL formed by combining the namespace URL defined in the XML namespace declaration (xmlns) with the local name (LocalPart) must be unique within the domain of the organization or person that named local name.

If the terminology indicated in Appendix A or B is used by the organization or individual, it might be used with additions or modifications. In that case, if terminology appears to have been redefined, even if the appendices were referenced, then the namespace URL may be decided within the organization or personal domain.

If the <name> element and key attribute are based only on the local name (LocalPart), then the XML namespace rules specify that the local name (LocalPart) be defined by the MaiML namespace for the value specified for the MaiML file. That could possibly hinder AI data analysis or other applications, but assuming the local name (LocalPart) is named by an organization or individual, the problem can be avoided by temporarily assigning a namespace URL value.

Strictly speaking, URL values can be either URL values with US-ASCII characters or IRI values with Unicode characters and UTF-8 encoding that can be used directly as they are. If naming rules for no-colon name types (xs:NCName) are restricted to Unicode basic Latin characters with UTF-8 encoding, then the URL value will satisfy URL, rather than IRI, specifications. That provides the side benefit of being easier to us in programming language environments that do not support IRI specifications.

Note that whether or not naming rules for relevant no-colon name types (xs:NCName) are restricted to basic Latin Unicode characters, URL values should be converted to UTF-8 byte strings when calculating UUID values using Version 3 or 5.

B.4 ID Type (xs:ID)

The ID type (xs:ID) is an XML modified name used for id attributes. They must satisfy the conditions for no-colon name types (xs:NCName) and also not duplicate element or attribute values for arbitrary ID types (xs:ID) in MaiML files.

According to the specifications, id attribute values for all elements in the MaiML file should be assigned a serial number, such as id00001, regardless of the element name. However, if there are multiple elements with the same no-colon name type (xs:NCName) (there are multiple instances with the same class in the original data) or if an <insertion> element is used to reference an external value, then formalizing a practice of using a character string (snake_case) formed by connecting the element name, no-colon name time (xs:NCName), and instance serial number with the low line character "_" (U+005F), for example, offers the benefit of making it easier to create a converter and easier for MaiML file users to add id attribute criteria to search algorithms.

B.5 IDREF Type (xs:IDREF)

The IDREF type (xs:IDREF) is an XML modified name used for ref attributes. They must satisfy the conditions for no-colon name types (xs:NCName) and also be a character string that is declared within arbitrary ID type (xs:ID) elements or attributes in MaiML files.

Index

This common data format guideline represents part of what has been achieved by a project of the Japanese Ministry of Economy, Trade and Industry.