JATS Compatibility Meta-Model Description
Draft 0.7

By:

B. Tommie Usdin, Mulberry Technologies, Inc.

Deborah A. Lapeyre, Mulberry Technologies, Inc.

Laura Randall, NCBI/NLM/NIH

Jeffrey Beck, NCBI/NLM/NIH

Draft of July 12, 2016

Executive Summary

Anyone who creates vocabularies based on ANSI/NISO Z39.96-2015 JATS will assume that the resulting documents will be compatible with existing JATS documents and the systems that manipulate them. This is not necessarily the case!

The goal of this document is to enable creators and maintainers of JATS-based document models to know when the extensions they make to JATS models are JATS-compatible and to suggest ways in which they can achieve their modeling goals in a JATS-compatible way.

JATS compatibility is evaluated on the element/attribute and tag set levels. A Structure in a JATS-based model that uses an existing JATS name must have the same semantic meaning as in JATS. Additionally, there are a number of “Properties” that a structure might or might not have. For example: an element might or might not be allowed to contain character data; an attribute might or might not be an XML ID or an XML IDREF; a structure might or might not have a recursive section-like model.

An element or attribute defined by a JATS extension is “JATS-compatible” if it has the same semantic meaning as the object of the same name in JATS and the object matches the corresponding JATS object on all of the Compatibility Properties identified in this document. A tag set that is an extension of JATS is “JATS-compatible” if all of the shared elements and attributes are JATS-compatible.

This document is intended to help developers of new JATS-related XML vocabularies create those vocabularies in ways that usefully *extend the reach* of the JATS vocabularies without *conflicting with* current JATS vocabularies. It documents those things that must not change about a model for it to be consistent with the JATS models.

We suggest that a JATS-family consistency board be formed to:

- Maintain these guidelines,
- Assist the developers of new JATS-family vocabularies in understanding and conforming to these guidelines,
- Add to these guidelines, and in particular add to the Compatibility Properties lists, as new structures are added to existing JATS vocabularies and/or as new vocabularies introduce new structures to the community.

1 Introduction

The JATS models, now ANSI/NISO Z39.96-2015, are the first of what we anticipate will be a growing set of related XML document models. JATS was designed to enable customization and extension. The JATS Standard describes new tag sets that are subsets of one of the JATS tag sets as “Conforming”, and new tag sets that extend the JATS vocabulary as “based on JATS”.

Documents that are encoded with conforming JATS vocabularies can generally be intermixed with documents that are encoded with JATS. For example, JATS-conforming documents can be processed with tools that format, search, and manage JATS documents.

It is reasonable that people creating vocabularies based on JATS will assume that they simply need to extend tools and databases that work with JATS and JATS-conforming documents to be able to use these tools with their new JATS-based documents. **This is not necessarily the case!** There are some ways in which the JATS models can be extended that will work gracefully in a JATS environment and other ways in which JATS could be extended that will create documents that are not likely to work well with existing JATS documents and systems tuned for them.

The goal of this document is to enable creators and maintainers of this growing set of document models to know when the extensions they make to JATS models are JATS-compatible, and to suggest ways in which they can achieve their modeling goals in JATS-compatible ways when they consider an option that is not JATS-compatible. These guidelines are intended to help vocabulary designers find a good balance between consistency and customization when interacting with these models or making new models based on existing models.

When creating a new JATS-based vocabulary, customization may be needed because both the documents and the expected uses of documents may differ from existing JATS-based vocabularies. Structures to encode the unique and/or important features of each document type are key to valuable XML encoding of those documents. Typically, the top level structure(s) of the documents and some subject-matter specific structures will be unique to a vocabulary. However, most of the structures of textual documents (internal structures such as figures, footnotes, lists, and citations) will be the same among all prose documents and should be the same in all documents in the JATS environment.

Consistency among the models is desirable because consistent models allow users to:

1. Use tools developed for one document type to be used with other document types in the family, or with new families also based on JATS, in such a way that adding processing to handle new features of the vocabularies does not require re-engineering the entire tool;
2. Interchange XML documents, document fragments, and metadata, not only between users of the same vocabulary, but among users of other vocabularies in the same family;
3. Manage and search the documents in the vocabularies as a group, for example, putting them all into a single XML-cognizant database and searching the entire collection;
4. Format the documents for print and electronic display using existing stylesheets and formatting applications adding only formats for new structures; and
5. Minimize the learning needed to add another document type in an environment where people already know one of the document types in the family.

2 Purpose

This document establishes what it means for a tag set to be JATS-compatible and provides guidance on how a tag set can be developed or modified to be JATS-compatible. It provides both the principles on which JATS compatibility can be decided and some specific criteria that must be met to ensure JATS compatibility.

This document is intended to help developers of new JATS-related XML vocabularies create those vocabularies in ways that usefully *extend the reach* of the JATS vocabularies without *conflicting with* current JATS vocabularies. It documents those things that must not change about a model for it to be consistent with the JATS models.

JATS compatibility is evaluated both on a tag set level and on an element-by-element, attribute-by-attribute basis. Some of the evaluation criteria are expressed as general Design Principles (see section “Design Principles”), and some are expressed as Compatibility Properties (see “Appendix B: JATS Compatibility Properties Catalog”).

In order for a tag set to be JATS-compatible, the tag set as a whole must meet all of the rules expressed in this document, and all of the Structures (individual element and attribute models) in the tag set must meet all of the Design Principles. Agreement on these Design Principles will ensure that the document models are fundamentally compatible. For example, while there are XML document models that prohibit any mixed content elements, such models are not JATS-compatible.

In addition to the general Design Principles, there are a number of “Properties” which a structure might or might not have. For example: an element might or might not be allowed to contain character data; an attribute might or might not be an XML ID or an XML IDREF; a structure might or might not have a recursive section-like model.

The JATS Compatibility Properties Catalog (Appendix B) shows the value of each Property for each individual element and attribute. In order to be JATS-compatible, each element and attribute must match the values of each of the Compatibility Properties shown in the Catalog.

If an element or attribute does not match previously cataloged JATS-based vocabularies on all of these Properties, the new vocabulary should re-name the element or attribute so that it is not in conflict with a similarly named pre-existing structure. These new structures can then be added to the JATS Compatibility Properties Catalog so that as subsequent vocabularies check for compatibility they will check against an ever-growing JATS Compatibility Properties Catalog. There is no intention to prohibit new vocabularies from adding as many new structures as needed. The intention is simply to prevent the same element or attribute name from being used in a way that is incompatible with existing JATS vocabularies. We want to encourage new vocabularies to use existing structure names when their use is JATS-compatible.

Example:

The JATS model for <name> allows it to contain only other elements, not characters. JATS <name> can contain <surname>, <given-names>, <prefix>, and <suffix> in a specified sequence. If another vocabulary wanted to change the sequence of the elements in <name>, or to add another element (perhaps <avatar>), this would be compatible with the JATS <name> model, and the new model would match on all of the Properties in the JATS Compatibility Properties Catalog.

Note: the sequence of child elements in an element is not constrained by these Design Principles and there is no “Matches Sequence” or “Matches Model” Property in the Catalog.

If another group wanted to put data characters directly inside <name>, they should be advised that this does not match the “whitespace handling” Property, and therefore is incompatible with the JATS model for <name>. A new element name would need to be created for the name variant. (Note: We hope someone would point them to JATS’ <string-name> element, which was created for just this purpose.)

For example, a JATS name might be:

```
<name><surname>McCrohan</surname>
<given-names>John</given-names><prefix>Capt.</prefix></name>
```

And we can imagine a JATS-compatible name such as:

```
<name><avatar>McThor</avatar><surname>McCrohan</surname><given-names>John
</given-names><prefix>Capt.</prefix></name>
```

But this would not be a compatible JATS <name>:

```
<name>Capt. <given-names>John</given-names><surname>McCohan</surname></name>
```

It is important that JATS element and attribute names be used in new JATS-compatible vocabularies when the meaning of the content is semantically similar to the meanings of previous JATS vocabularies and the model matches on all of the Compatibility Properties. Reusing existing structures will make it possible to extend existing JATS tools for new document sets instead of developing new tools and will make it practical to search across a body of JATS-compatible documents.

Example:

There might be a document set in which it is important to identify the types of organizations and the specific organizations that host each of the events being described. If such a vocabulary were not intending to be JATS-compatible, it might model these as <org-type>, <educational>, <not-for-profit>, <commercial>. We can imagine an event tagged as:

```
<event><name>…</name><date>…</date><location>…</location>...
<sponsor><educational><university>University of Frostbite Falls
</university>
<department>Dept of Moose and Squirrel Security
</department></educational></sponsor></event>
```

However, JATS has structures for encoding such organization names, and we encourage new vocabularies to use them. In a JATS-compatible vocabulary we suggest this information should be encoded as:

```
<event><name>…</name><date>…</date><location>…</location>...
<sponsor>
<institution content-type="edu">University of Frostbite Falls</institution>
<institution content-type="dept">Dept of Moose and Squirrel
Security</institution>
</sponsor></event>
```

3 Application

It is in the best interests of both the users of existing JATS-family vocabularies and the users of new JATS-family vocabularies if existing vocabularies remain interoperable as they evolve and if new vocabularies are interoperable with the existing JATS-family vocabularies.

This document describes a number of checks that need to be made in order to ensure that a vocabulary is JATS-compatible. Some of these checks can be partially automated and a few can be fully automated, but most will require human judgment. Software could easily check to ensure that attributes identified on the Attributes Type Property as being of type ID are of type ID and attributes identified as being of type IDREF are of type IDREF or IDREFS. It is relatively easy to check to see that the models of elements identified as section-like have section-like models or models that are subsets of the section model. However, only a person can check to see if the meaning of an element in a new vocabulary is semantically similar to the meaning of the element in the JATS vocabulary.

In order to facilitate this interaction, we suggest that a JATS-family consistency board be formed.

This board will be tasked with:

- Maintaining these guidelines,
- Assisting the developers of new JATS-family vocabularies in understanding and conforming to these guidelines,
- Adding to these guidelines and, in particular, adding the appropriate values for each new element or attribute to each of Properties as new structures are added to existing JATS vocabularies and/or as new vocabularies introduce new structures to the community.

4 Design Principles

These Design Principles are intended to enable people to make JATS-based tag sets fit together gracefully in ways that are easier to describe in prose than in a checklist.

4.1 Element and Attribute Names

4.1.1 Semantic Match

The most important rule for determining if a proposed element or attribute already exists in JATS is a requirement for semantic equivalence. If the definition/meaning of an element or attribute is different from that of the JATS structure, the tag set will not be JATS-compatible unless that structure is given a different element or attribute name. It is not JATS-compatible to repurpose a JATS structure for information with a different meaning even if that might create an appropriate display and even if the new structure has the same model as the one already in JATS. Developers must use the JATS definitions to establish meaning; name equivalences may be misleading.

Note that while in natural language it is common for words to have multiple meanings which might be significantly different, replicating this in tag sets can cause significant difficulty with interchange and search. If a meaning seems distantly related to that of an existing JATS structure, or if it seems that the meaning of an existing structure is being stretched to include the meaning needed in a new context, we urge the developers to create a new structure instead.

Repeated stretching of the meanings of markup structures seriously degrades the value of the markup.

Example:

A vocabulary that has no need for the JATS <term> element, which is defined as “Word, phrase, graphic, chemical structure, equation, or other noun that is being defined or described.” but would find it useful to have an element for “Length of time for which an office holder is elected” must not use <term> for this information if it wants to be JATS-compatible; a newly named element (such as <term-length>) is required in order for this new vocabulary to join the JATS models.

4.1.2 Creating New Objects

The same name may only be used for a structure if it matches all of the Compatibility Properties in the JATS Compatibility Properties Catalog. Our expectation is that if a modified model matches the JATS model on the these Properties, this model can, and should, use the same name JATS uses for that structure. A paragraph is a paragraph is a paragraph, even if we do not agree 100% on what can be inside one or redefine it slightly.

If, however, a modified model *does not match* on even one of the Compatibility Properties, it should be given a different name—one that is not already in JATS. This rule is not intended to restrict the structures or models that can be created in the JATS environment; this is simply a mechanism to differentiate between those variations that are comfortably compatible and those that should be differentiated as something new.

Example:

Paragraph (<p>) in JATS has a mixed content model. If a vocabulary wanted paragraphs that contained labels, titles, text-blocks, and perhaps nested paragraphs, that model in the JATS environment would have to be named something other than Paragraph (<p>).

Similarly, there is no restriction on the addition of attribute values, either in the model or documented as suggested values provided the new values do not redefine existing values. New values of an attribute must match all of the applicable Properties in the JATS Compatibility Property Catalog.

JATS Naming Conventions

Although it is not a requirement for compatibility, we recommend that new elements and attributes follow the JATS naming conventions. These conventions are documented in the JATS Tag Library (https://jats.nlm.nih.gov/archiving/tag-library/1.1/chapter/imple-naming-rules.html)

4.2 Proper Subsetting is Always Allowed

4.2.1 Subsetting Elements

Changing a model to be a subset of that JATS model is always JATS-compatible. Element models may be subsets, even very small subsets, of the JATS structure.

Example:

In the JATS Compatibility Properties Catalog, <abstract> is a section-like model. For example, the model for <abstract> in JATS Publishing 1.1 is:

- Label <label>, optional, followed by
- Title <title>, optional, followed by
- Paragraph <p>, optional repeatable, followed by
- Section <sec>, optional repeatable.

If a vocabulary wants an <abstract> model that does not allow internal sections, such as:

- Label <label>, optional, followed by
- Title <title>, optional, followed by
- Paragraph <p>, optional repeatable

that would be JATS-compatible because the model is a subset of the section-like model. In fact, it would be JATS-compatible if the new model for abstract were as small a subset as

- Title <title>, optional, followed by
- Paragraph <p> (one or more, required)

or even

- Paragraph <p> (one and only one)

4.2.2 Making Attribute Values More Restrictive

Attribute values may be more specific than those in existing JATS vocabularies provided documents using the new model would be valid according to the restrictions in the JATS Compatibility Properties Catalog. For example, removing values from a list of attribute values is JATS-compatible, as is providing a list of values where the previous model is CDATA. (It is not acceptable to provide a list of values for an attribute of type ID, because this is not more restrictive, it is of a different type.) Suggested values (in the JATS Tag Libraries) are simply that, so using values other than the suggested values is also JATS-compatible.

Example:

@abstract-type is a #CDATA value attribute in JATS with a suggested value list of more than a dozen values in the non-normative documentation. If a vocabulary wanted to restrict @abstract-type to “short”, “graphical”, and “audio”, that would be JATS-compatible.

4.3 Pointing (Linking): from the Many to the One

Linking and pointing within documents is used for many purposes including cross-references between locations in the text, associating parts of the text with references, and for linking contributors with their affiliations. A wide variety of systems for associating one part of a document with another is possible in XML. Interoperability of XML documents is greatly enhanced if the same system of location identifiers and pointers is used in all of the documents in the environment.

In JATS, the structure that occurs only once is assigned an identifier ID (an attribute of XML type ID), and it is pointed to from the type of structure that might occur more than once, such as a cross-reference.

Example:

Each bibliographic reference has an ID (attribute @id), and each cross-reference to that bibliographic reference uses the value of that ID in its IDREF (attribute @rid).

Each organizational affiliation in the article metadata of the article metadata may take an ID, and each of the contributors associated with that organization have a link (using an attribute of type IDREF) on a cross-reference <xref> to that affiliation.

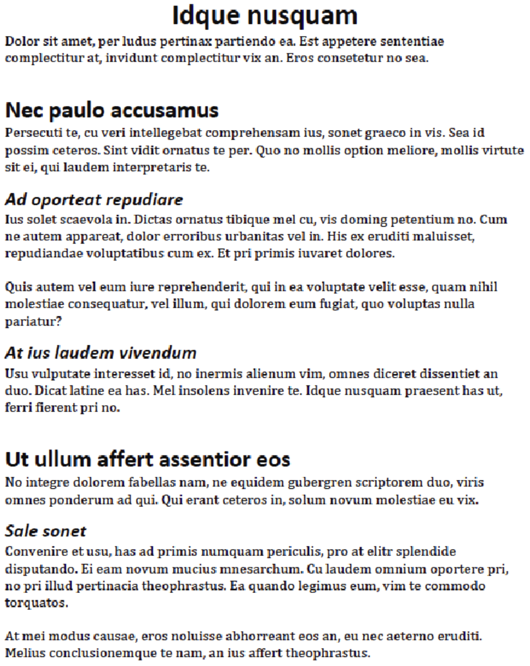
4.4 Section-based Logic of a JATS Document

A fundamental characteristic of complex prose documents is that they have headings with multiple styles that imply a hierarchical structure and have text subsidiary to them.

There are many ways to model this. Word processors typically have multiple levels of headings intermixed with paragraphs and other text structures. Some systems have mechanisms to identify sections and subsections, associating the paragraphs “below” a heading with that heading. Some nest the headed sections, with lower-level sections appearing inside higher-level sections. Some number each level and control the levels of nesting allowed while others allow sections to occur inside sections creating a recursive model.

The recursive section model:

A document that looks like this:



Idque nusquam

Dolor sit amet, per ludus pertinax partiendo ea. Est appetere sententiae complectitur at, invidunt complectitur vix an. Eros consetetur no sea.

Nec paulo accusamus

Persecuti te, cu veri intellegebat comprehensam ius, sonet graeco in vis. Sea id possim ceteros. Sint vidit ornatus te per. Quo no mollis option meliore, mollis virtute sit ei, qui laudem interpretaris te.

Ad oporteat repudiare

Ius solet scaevola in. Dictas ornatus tibique mel cu, vis doming petentium no. Cum ne autem appareat, dolor erroribus urbanitas vel in. His ex eruditi maluisset, repudiandae voluptatibus cum ex. Et pri primis iuvaret dolores.

Quis autem vel eum iure reprehenderit, qui in ea voluptate velit esse, quam nihil molestiae consequatur, vel illum, qui dolorem eum fugiat, quo voluptas nulla pariatur?

At ius laudem vivendum

Usu vulputate interesset id, no inermis alienum vim, omnes diceret dissentiet an duo. Dicat latine ea has. Mel insolens invenire te. Idque nusquam praesent has ut, ferri fierent pri no.

Ut ullum affert assentior eos

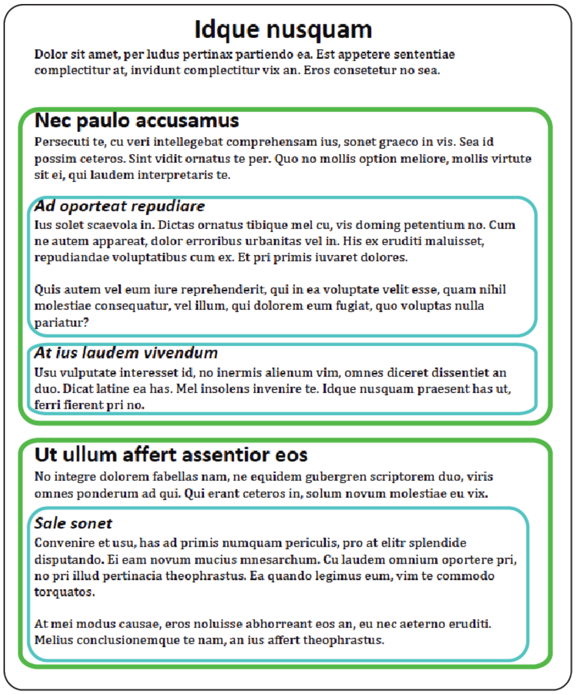
No integre dolorem fabellas nam, ne equidem gubergren scriptorem duo, viris omnes ponderum ad qui. Qui erant ceteros in, solum novum molestiae eu vix.

Sale sonet

Convenire et usu, has ad primis numquam periculis, pro at elitr splendide disputando. Ei eam novum mucius mnesarchum. Cu laudem omnium oportere pri, no pri illud pertinacia theophrastus. Ea quando legimus eum, vim te commodo torquatos.

At mei modus causae, eros noluisse abhorreant eos an, eu nec aeterno eruditi. Melius conclusionemque te nam, an ius affert theophrastus.

Would be modeled in JATS as nested sections:



Idque nusquam

Dolor sit amet, per ludus pertinax partiendo ea. Est appetere sententiae complectitur at, invidunt complectitur vix an. Eros consetetur no sea.

Nec paulo accusamus

Persecuti te, cu veri intellegebat comprehensam ius, sonet graeco in vis. Sea id possim ceteros. Sint vidit ornatus te per. Quo no mollis option meliore, mollis virtute sit ei, qui laudem interpretaris te.

Ad oporteat repudiare

Ius solet scaevola in. Dictas ornatus tibique mel cu, vis doming petentium no. Cum ne autem appareat, dolor erroribus urbanitas vel in. His ex eruditi maluisset, repudiandae voluptatibus cum ex. Et pri primis iuvaret dolores.

Quis autem vel eum iure reprehenderit, qui in ea voluptate velit esse, quam nihil molestiae consequatur, vel illum, qui dolorem eum fugiat, quo voluptas nulla pariatur?

At ius laudem vivendum

Usu vulputate interesset id, no inermis alienum vim, omnes diceret dissentiet an duo. Dicat latine ea has. Mel insolens invenire te. Idque nusquam praesent has ut, ferri fierent pri no.

Ut ullum affert assentior eos

No integre dolorem fabellas nam, ne equidem gubergren scriptorem duo, viris omnes ponderum ad qui. Qui erant ceteros in, solum novum molestiae eu vix.

Sale sonet

Convenire et usu, has ad primis numquam periculis, pro at elitr splendide disputando. Ei eam novum mucius mnesarchum. Cu laudem omnium oportere pri, no pri illud pertinacia theophrastus. Ea quando legimus eum, vim te commodo torquatos.

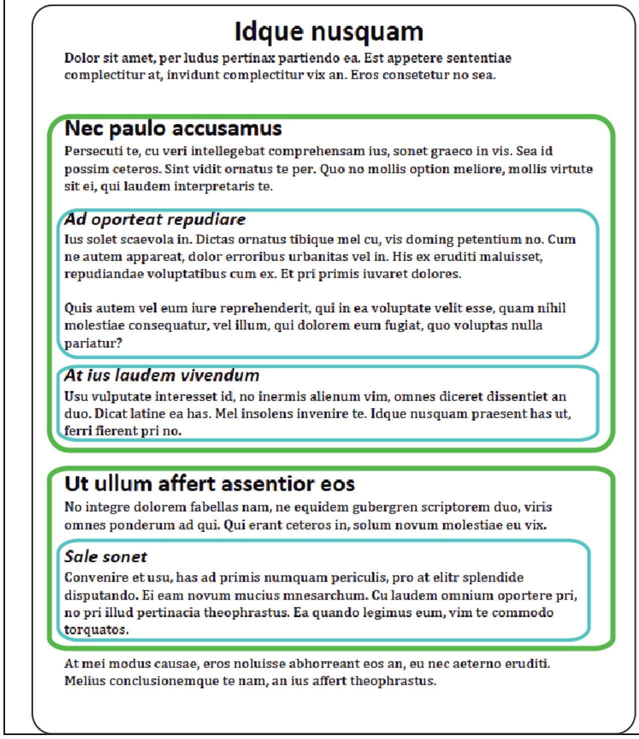
At mei modus causae, eros noluisse abhorreant eos an, eu nec aeterno eruditi. Melius conclusionemque te nam, an ius affert theophrastus.

JATS sections are infinitely recursive. That is, a section may contain sections, which may contain sections without limit. (If a JATS-compatible application needs to limit the nesting of sections for some reason, this might be done using a mechanism other than the XML model, for example, a Schematron that implements these business rules.)

4.4.1 Extent of a Section

JATS elements that are Sections or modeled like sections do not allow block-like objects (such as paragraphs) to occur after the Sections start, unless they are clearly identified as Section-Tail objects. If block-like objects were allowed here, readers of the documents would be confused.

If the last paragraph of the document illustrated above were not in the Sections it follows the reader would have no way to know this:



Idque nusquam

Dolor sit amet, per ludus pertinax partiendo ea. Est appetere sententiae complectitur at, invidunt complectitur vix an. Eros consetetur no sea.

Nec paulo accusamus

Persecuti te, cu veri intellegebat comprehensam ius, sonet graeco in vis. Sea id possim ceteros. Sint vidit ornatus te per. Quo no mollis option meliore, mollis virtute sit ei, qui laudem interpretaris te.

Ad oporteat repudiare

Ius solet scaevola in. Dictas ornatus tibique mel cu, vis doming petentium no. Cum ne autem appareat, dolor erroribus urbanitas vel in. His ex eruditi maluisset, repudiandae voluptatibus cum ex. Et pri primis iuvaret dolores.

Quis autem vel eum iure reprehenderit, qui in ea voluptate velit esse, quam nihil molestiae consequatur, vel illum, qui dolorem eum fugiat, quo voluptas nulla pariatur?

At ius laudem vivendum

Usu vulputate interesset id, no inermis alienum vim, omnes diceret dissentiet an duo. Dicat latine ea has. Mel insolens invenire te. Idque nusquam praesent has ut, ferri fierent pri no.

Ut ullum affert assentior eos

No integre dolorem fabellas nam, ne equidem gubergren scriptorem duo, viris omnes ponderum ad qui. Qui erant ceteros in, solum novum molestiae eu vix.

Sale sonet

Convenire et usu, has ad primis numquam periculis, pro at elitr splendide disputando. Ei eam novum mucius mnesarchum. Cu laudem omnium oportere pri, no pri illud pertinacia theophrastus. Ea quando legimus eum, vim te commodo torquatos.

At mei modus causae, eros noluisse abhorreant eos an, eu nec aeterno eruditi. Melius conclusionemque te nam, an ius affert theophrastus.

4.4.2 Parts of a Section

The JATS section-like model has four parts, all of which are optional but which must occur in this sequence if they occur:

- Section-Head
- Section-Blocks
- Sections
- Section-Tail

Note: These are not element names; they are roles that elements may play in the structure of a section. In any particular section or section-like model, each of these roles may be played by any number of elements, or none.

The parts of the section-model are illustrated in Appendix A: Parts of a Section, page 17.

Some section-like structures, e.g., <sec>, <abstract>, <boxed-text>, and <app>, allow all parts of the section model. Others, e.g., <statement>, contain only part of the section model.

Because the rules of XML modeling prohibit non-deterministic models[1]*, in most cases a structure may not be allowed in more than one of these roles in a specific section-like model. If a structure that is allowed to play a role in the section model is needed in another part of the section it must be put in a containing element so that the model remains deterministic (which some tools call unambiguous).*

Example:

In JATS Sections <sec>, reference lists <ref-list> is a Section-Tail structure. That is, it may occur at the end of a Section after any nested structures. However, it may not occur in the Section-Blocks area of the Section as a definition-list might.

4.4.3 Section-Head

The Section-Head contains identification of the Section and descriptions of it. This may include labels, titles, abstracts, keywords, licensing and permissions information, and/or a grouping of such information such as <sec-meta>.

Example:

In JATS, <bio> has a section-like model, but does not include <object-id>. It would be JATS-compatible to add <object-id> to the Section-Head area of the <bio> model because it is an identifier.

In JATS, the Section-Head of <bio> includes <sec-meta>, <label>, and <title>. Since the Section-Head is optional in the section model, all of these could be removed in a JATS-compatible vocabulary, leaving <bio> to contain only Section-Blocks, Sections, and Section-Tail.

4.4.4 Section-Blocks

The blocks in a Section or section-like structure appear before the first of the subsections. The most common blocks are paragraphs, tables, and figures, but many other block-level structures may be allowed. The blocks must appear only *before* the first of the nested sections because it is the habit of readers to assume that any paragraphs (or other blocks) that appear after a heading are part of the section identified by that heading. If a section were to end and be followed by a block, the reader would assume the block was inside that section.

[1] See the XML Specification, and specifically section 3.2.1 Element Content at https://www.w3.org/TR/REC-xml/#sec-element-content.

Example:

Reference lists <ref-list>s are not allowed in the Section-Blocks area of Sections <sec> in JATS; they are allowed only in the Section-Tail area of JATS sections. It would be possible, and reasonable, to make a JATS-compatible model of <sec> that allows <ref-list>s in the Section-Blocks area of <sec>s, but this would mean that in this model <ref-list>s would not be allowed in the Section-Tail area of <sec>s.

4.4.5 Sections

JATS uses a recursive nested model for sections. Section levels are computed from the location on the Section in the document, heading styles are computed by the nesting level of the Section in which they are contained, and Sections may nest as deeply as needed.

Example:

Sections may nest as deeply as needed in a document:

```
<sec><label>2</label><title>Pets</title>
  <sec><label>2.1</label><title>House Pets</title>
    <sec><label>2.1.1</label><title>Furry House Pets</title>
      <sec><label>2.1.1.1</label><title>Dogs</title>
        <sec><label>2.1.1.1.1</label><title>Lap Dogs</title>
          <sec><label>2.1.1.1.1.1</label><title>Shih Tzu</title>
          </sec>
        </sec>
      </sec>
    </sec>
  </sec>
</sec>
```

4.4.6 Section-Tail

The Section-Tail is information that may come at the end of a Section after the end of the last nested Section. Section-Tail structures are typically structures that may be referenced from inside the Section: <notes>, <fn-group>, <glossary>, and <ref-list>.

Example:

It is common in JATS documents for sections to end with reference lists. It would be a reasonable extension for a JATS-compatible vocabulary to add test questions to the tail of sections containing instructional material.

5 Compatibility Properties

The JATS Compatibility Properties Catalog (Appendix B) summarizes several Properties on which a new use must be compared to the existing JATS uses. The Property list can be thought of as a checklist. In order for a proposed structure to match an existing JATS structure, it must have the same values on each of the applicable Properties. Some of these Properties apply to all structures (Element versus Attribute), some are relevant only to some structures (Attribute Type applies only to attributes). Some are yes/no, where an "X" means yes and no value means no (for example, is the element an alternatives wrapper element), and at least one Properly (Whitespace Handling Type) has three options which are represented by single-letter codes.

Note: The Compatibility Properties described here and summarized in Appendix B are far from all of the possible properties of a tag set. It would be possible, for example, to identify which structures in a tag set are pointers to content outside the current document, which are expected to have numerical content, and which are metadata about the document. These are properties of the tag set, but are not included in the Compatibility Properties list because matching on them is not necessary to achieve the types of document interoperability anticipated by this document.

5.1 Element Versus Attribute

If JATS has put certain content within an element, as narrative text or a child element, a new JATS vocabulary should not make that content into attributes. If the content must be made into an attribute, a new element to hold this attribute must be created. Similarly, if JATS defines the content as an attribute value, a new vocabulary should not make it into child elements; a new and different structure with child elements would need to be created.

Further, although JATS has several examples of the same name being used for an element and an attribute, (e.g., ‘corresp’ and ‘version’) we strongly discourage this practice.

Example:

In JATS, keywords are tagged as elements and describe the content at the level in which they appear (inside a keyword-group inside a larger structure such as an article or a section). Keywords in the article metadata describe the whole article; keywords inside an appendix describe the appendix; keywords inside a figure describe that figure. It would be possible for a vocabulary to put an attribute on various structures and put the keywords into that attribute. Since this is the same information contained in the JATS <kwd-group> and <kwd> elements, we encourage other JATS vocabularies to put their keywords into elements as JATS does. Should there be a good reason that the other vocabulary use an attribute to contain this information, the attribute should not be given a name already in use as either an element or an attribute name.

5.2 Whitespace Handling Type

This apparently obscure detail of the way XML is processed can have significant impact on the ability to re-use tools among documents in a heterogeneous collection. If all of the documents in the collection do not have the same whitespace handling rules for elements, there may be unfortunate (and in some cases misleading) effects on the display of the document content.

Whitespace handling in XML:

By definition in XML documents, there are two types of whitespace: significant whitespace and insignificant whitespace. In any context where the whitespace is insignificant, any process touching the XML document (such as an XSLT transformation or an XML editor) may remove all whitespace, add whitespace, or collapse multiple whitespaces into one at any time. In a context where whitespace is significant, any process touching the XML document may collapse multiple whitespaces into one or add additional whitespace where whitespace already occurs, but it *may not remove* all whitespace or insert whitespace where there was none in the document.

In XML all whitespace characters are considered identical: spaces, tabs, line-feeds, and carriage-returns are treated as equivalent.

Element-like whitespace

Content models that contain only elements (no characters) have insignificant whitespace. By definition, the following XML fragments (if modeled using JATS, which defines all of these elements except <kwd> as having element content) are identical, and a process could convert one to another at any time and no XML processor is allowed to treat these three examples differently:

```
<kwd-group specific-use="mobile-nav">
  <nested-kwd>
    <kwd>dosing</kwd>
    <nested-kwd>
      <kwd>geriatric</kwd>
      <nested-kwd>
        <kwd>Heart failure</kwd>
      </nested-kwd>
    </nested-kwd>
  </nested-kwd>
</kwd-group>

<kwd-group specific-use="mobile-nav">
  <nested-kwd><kwd>dosing</kwd>
    <nested-kwd><kwd>geriatric</kwd>
      <nested-kwd><kwd>Heart failure</kwd>
      </nested-kwd>
    </nested-kwd>
  </nested-kwd>
</kwd-group>

<kwd-group specific-use="mobile-nav"><nested-kwd><kwd>dosing</kwd>
<nested-kwd><kwd>geriatric</kwd><nested-kwd><kwd>Heart failure</kwd>
</nested-kwd></nested-kwd></nested-kwd></kwd-group>
```

Data-like whitespace

Content models that allow character data or mixed content have significant whitespace. By definition, the following XML fragments are identical, and a process could convert one to another at any time and no XML processor is allowed to treat these three examples differently:

```
<aff id="StLukes">Department of Health Care for the
Elderly, St Luke&#x2019;s Hospital, Bradford BD5 0NA</aff>

<aff id="StLukes">Department of Health Care
                  for the Elderly,
                  St Luke&#x2019;s Hospital,
                  Bradford BD5 0NA</aff>

      <aff id="StLukes">Department
                    of
                      Health Care
                         for the Elderly,
                  St Luke&#x2019;s
                      Hospital,
                         Bradford
                              BD5
                                 0NA</aff>
```

An XML processor may not convert such content, if it has significant whitespace, into either of the following (whitespace is added in the first and removed in the second):

```
<aff id="StLukes">
     Department of Health Care for the Elderly, St Luke&#x2019;s Hospital, Brad
     ford BD5 0NA</aff>

<aff id="StLukes">DepartmentofHealthCareforthe
Elderly,StLuke&#x2019;sHospital,BradfordBD50NA</aff>
```

Preserved whitespace

Whitespace folding (collapsing each whitespace node[2] into a single space character (U+0020)) is the default expectation in XML that has data-like whitespace handling. In JATS we have one other way to handle whitespace. It may be specified (using “xml:space="preserve"”) that all whitespace is to be preserved. For elements where whitespace is to be preserved, the processor may not insert whitespace, may not collapse whitespace, and may not convert one whitespace character to another. This is typically used in examples like this:

```
<code code-type="dtd"  xml:space="preserve">
&lt;!ELEMENT  implications  (tree+) &gt;
&lt;!ELEMENT  tree          (root, branches) &gt;
&lt;!ELEMENT  root          (term, synonym?) &gt;
&lt;!ELEMENT  branches      (term | (term, synonym) | tree)* &gt;
</code>

or

<preformat preformat-type="dialog">
C:\users\lap make
  'make' is not recognized as:
      - an internal or external command
      - an operable program
      - a batch file
</preformat>
```

The Whitespace Handling Property identifies each element as having one of three types of whitespace handling:

Element-like whitespace handling (whitespace is insignificant): This is the way whitespace is handled in element-only content models. Processors may remove all whitespace and add one or more whitespace characters in this context (that is, between the elements contained in this context) without making any difference in how the content is processed. In Appendix B, “Element-like whitespace handling” is represented by “E”.

Data-like whitespace handling (whitespace is significant and may be folded): This is for content models that allow character content, that have mixed content, or that are declared to be empty. In these contexts, a processor may collapse multiple whitespaces into one whitespace, convert all kinds of whitespace into spaces, and may add whitespace any place where there is already a whitespace character. (Empty elements are specified to have whitespace handling of type D because it does not affect JATS-compatibility if empty elements are converted to allow character data content.) In Appendix B, “Data-like whitespace handling” is represented by “D”.

Preserve whitespace (whitespace is to be preserved exactly as it is in the document): No whitespace may be added, none removed, and none converted from one type of whitespace to another. In Appendix B, “Preserve whitespace” is represented by “P”.

[2] A whitespace node is a string of one or more whitespace characters: carriage returns (U+000D), tabs (U+0009), spaces (U+0020), and line-feeds (U+000A).

Example:

<break/> is an empty element, and thus of type “D”.

<label> can contain characters (e.g., <label>Fig. 4</label>) and is of type “D”.

In the example:
   <string-name><surname>DerSimonian</surname>, <given-names>R</given-names>
   </string-name>
<string-name>, <surname>, and <given-names> are of type “D”.
(Note: The element <name> is of type “E”.)

Models of type “E” (Element-container) allow only other elements inside them. Any whitespace in an element of type “Element-container” is, by definition, insignificant. XML processors may ignore whitespace in these elements, may (and sometimes do) add whitespace (for example, to “pretty print” an XML file), and may (and sometimes do) remove all whitespace here.

Example:

A JATS <verse-group> contains one or more <verse-lines>, thus <verse-group> is of type “E”.

```
<verse-group>
<verse-line>Shall I compare thee to a summer's day?</verse-line>
<verse-line>Thou art more lovely and more temperate:</verse-line>
<verse-line>Rough winds do shake the darling buds of May,</verse-line>
<verse-line>And summer's lease hath all too short a date:</verse-line>
</verse-group>
<verse-group>
<verse-line>Sometime too hot the eye of heaven shines,</verse-line>
<verse-line>And often is his gold complexion dimm'd,</verse-line>
<verse-line>And every fair from fair sometime declines,</verse-line>
<verse-line>By chance, or nature's changing course, untrimm'd.</verse-line>
</verse-group>
```

While it would be possible to model verse as characters that include interleaved live breaks, this is not the way verse is modeled in JATS. Such a model of <verse-group> would be of type “D”, and not be JATS-compatible:

```
<verse-group>
Shall I compare thee to a summer's day?<br/>
Thou art more lovely and more temperate:<br/>
Rough winds do shake the darling buds of May,<br/>
And summer's lease hath all too short a date:<br/>
<br/>
Sometime too hot the eye of heaven shines,<br/>
And often is his gold complexion dimm'd,<br/>
And every fair from fair sometime declines,<br/>
By chance, or nature's changing course, untrimm'd.
</verse-group>
```

5.3 Section-like Property

While JATS has only one “section” model (<sec>), it has many section-like models, including <abstract>, <ack>, <app>, and <bio>. All of these structures consist of a Section-Head (identification and labeling information), optional repeating Section-Block structures, and optional repeating Sections; most have optional Section-Tails.

In order to be JATS-compatible, a model of one of the elements identified as section-like must have a section model or a model that is a subset of the section model. This is because a subset of a model is always JATS-compatible with the model. So, if a structure has a section-like model in the JATS Compatibility Properties Catalog, and a tag set models that element as having a subset of the section-like model, the new model is JATS-compatible.

Example:

In JATS, Boxed Text <boxed-text> has a section-like model. The boxed text model in a vocabulary that modeled boxed text as a title followed by one or more paragraphs would be JATS-compatible.

5.4 Alternatives Type Property

In JATS, alternatives elements contain several variations or versions *of the same content* that are intended to be used in different renderings or situations. Alternatives may be used to provide a name in more than one character set or to provide a mathematical expression as a graphic, as MathML, and perhaps as plain text. To the extent possible, each alternative represents the same information. Typically, only one of the alternatives is displayed, although in the cases of names more than one may be displayed if it is clear that they are alternate versions of the name of the same person and not multiple people.

JATS alternatives elements include <alternatives> (which is used for graphical or controlled-presentation content) and some alternatives elements with more specific uses, such as <citation-alternatives>, <name-alternatives>, <aff-alternatives>, and <collab-alternatives>.

These alternatives elements are merely containers, which have no meaning of their own, but contain a set of processing alternatives in such a way as to make it clear that all their children represent the same object. Alternatives is a Compatibility Property because an element that performs this function should perform no other (such as containing a label, title, or content).

5.5 Attribute of Type ID or IDREF

This Property indicates whether an attribute is an XML ID or XML IDREF.

In JATS, every element has either an optional or a required ID attribute, so any element in a document may be assigned an ID (enabling any element in the document to be uniquely identified so that it can, for example, be pointed to by cross-references). In JATS, many elements have an IDREF/IDREFS attribute, which means that many elements may point to other elements in the documents. It is JATS practice to point in only one direction, from the item that may occur many times to the item that may occur only once. Equally important, it is JATS best practice not to put reciprocal pointers into the XML documents.

Example:

It is common for a table to have many footnotes, some of which may be referenced more than once. Each of the table footnotes is identified with an ID (attribute @id), and each of the places that references that footnote is identified with an <xref> with an IDREF (attribute @rid) that points to the footnote. There is no IDREFS on the footnote that point to the places it was referenced.

In the JATS model for the one-to-many relationships of ID and IDREF, the thing that occurs only once is assigned an ID (in an attribute of type ID), and it is pointed to from a structure that might occur more than once (using an attribute of type IDREF). In order to maintain this consistency it is important that elements that have ID-type attributes in JATS have ID-type attributes in all vocabularies in the environment, and that elements that have IDREF- or IDREFS-type attributes use them consistently. Changing from an IDREF to IDREFS or vice versa does not affect JATS-compatibility because our goal is to be consistent with what is doing the pointing and what is being pointed at. Changing an attribute type from type ID to type CDATA would create interoperability problems because the tools that rely on IDs being unique within the document and structured according to the rules for IDs (which are much more limiting than the rules for CDATA) might be unable to correctly process that attribute with values that are allowed in CDATA.

Example:

It is common for journal articles to have multiple authors, for authors to have multiple affiliations, and for several authors of an article to be affiliated with the same organization. In recording the relationship between contributors and affiliations, each affiliation is given an ID and there are cross-references from the author/contributors to their affiliations. The affiliation elements do not point to the contributors.

```
<contrib-group>
<contrib contrib-type="author">
<name><surname>Forster</surname><given-names>Anne Williams</given-names></name>
<role>research physiotherapist</role>
<xref ref-type="aff" rid="StLukes"/>
<xref ref-type="aff" rid="RoyalInf"/>
</contrib>
<contrib contrib-type="author">
<name><surname>Young</surname><given-names>John G.</given-names></name>
<role>consultant physician</role>
<xref ref-type="aff" rid="RoyalInf"/>
</contrib>
<aff id="StLukes">Department of Health Care for the
Elderly, St Luke&#x2019;s Hospital, Bradford BD5 0NA</aff>
<aff id="RoyalInf">Academic Section of Geriatric
Medicine, Royal Infirmary, Glasgow G4 0SF</aff>
</contrib-group>
```

6 Determining JATS Compatibility

6.1 Compatibility of Individual Elements and Attributes

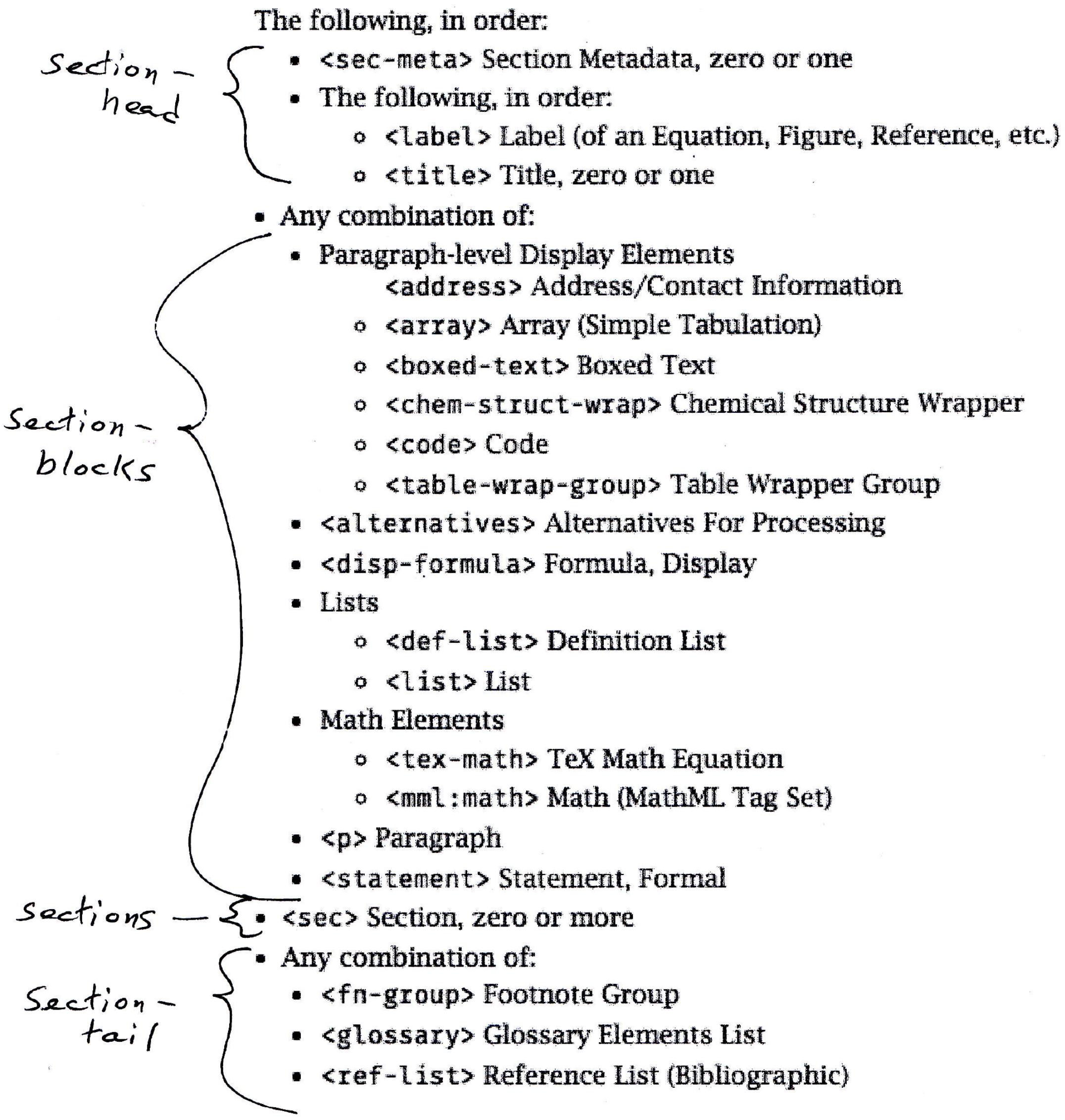
An element or attribute defined by a JATS extension is “JATS-compatible” if it has the same semantic meaning as the object of the same name in JATS (see 3.1 “Semantic Match”) and the object matches the corresponding JATS object on all of the Compatibility Properties.

6.2 Compatibility of New Tag Suites

A tag set that is an extension of JATS is “JATS-compatible” if all of the shared elements are JATS-compatible.

Appendix A: Parts of a Section

The JATS section-like model has four parts, all of which are optional but which must occur in this sequence if they occur: Section-Head, Section-Blocks, Sections, and Section-Tail. A simplified example is diagrammed below:


Section – head
Section – blocks
sections –
Section – tail

The following, in order:

- <sec-meta> Section Metadata, zero or one
- The following, in order:
  - <label> Label (of an Equation, Figure, Reference, etc.)
  - <title> Title, zero or one

- Any combination of:
  - Paragraph-level Display Elements
    <address> Address/Contact Information
    - <array> Array (Simple Tabulation)
    - <boxed-text> Boxed Text
    - <chem-struct-wrap> Chemical Structure Wrapper
    - <code> Code
    - <table-wrap-group> Table Wrapper Group
  - <alternatives> Alternatives For Processing
  - <disp-formula> Formula, Display
  - Lists
    - <def-list> Definition List
    - <list> List
  - Math Elements
    - <tex-math> TeX Math Equation
    - <mml:math> Math (MathML Tag Set)
  - <p> Paragraph
  - <statement> Statement, Formal
- <sec> Section, zero or more
- Any combination of:
  - <fn-group> Footnote Group
  - <glossary> Glossary Elements List
  - <ref-list> Reference List (Bibliographic)

Appendix B: JATS Compatibility Properties Catalog

APPENDIX B: JATS COMPATABILITY PROPERTIES CATALOG

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
abbr	attribute				
abbrev	element			D	
abbrev-journal-title	element			D	
abbrev-type	attribute				
abstract	element		X	E	
abstract-type	attribute				
access-date (deprecated)	element			D	
ack	element		X	E	
addr-line	element			D	
address	element			E	
aff	element			D	
aff-alternatives	element	X		E	
ali:free_to_read	element			D	
ali:license_ref	element			D	
align	attribute				
alt	attribute				
alt-text	element			D	
alt-title	element			D	
alt-title-type	attribute				
alternatives	element	X		E	
annotation	element			E	
anonymous	element			D	
app	element		X	E	
app-group	element		X	E	
arrange	attribute				
array	element			E	
article	element			E	
article-categories	element			E	
article-id	element			D	
article-meta	element			E	
article-title	element			D	
article-type	attribute				
assigning-authority	attribute				
attrib	element			D	
authenticated	attribute				
author-comment	element			E	
author-notes	element			E	
award-group	element			E	
award-id	element			D	
award-type	attribute				

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
axis	attribute				
back	element			E	
baseline-shift	attribute				
bio	element		X	E	
body	element		X	E	
bold	element			D	
border	attribute				
boxed-text	element		X	E	
break	element			D	
calendar	attribute				
caption	element			E	
cellpadding	attribute				
cellspacing	attribute				
chapter-title	element			D	
char	attribute				
charoff	attribute				
chem-struct	element			D	
chem-struct-wrap	element		X	E	
citation-alternatives	element	X		E	
city	element			D	
code	element			P	
code-type	attribute				
code-version	attribute				
col	element			D	
col-group	element			E	
collab	element			D	
collab-alternatives	element	X		E	
collab-type	attribute				
colspan	attribute				
comment	element			D	
compound-kwd	element			E	
compound-kwd-part	element			D	
compound-subject	element			E	
compound-subject-part	element			D	
conf-acronym	element			D	
conf-date	element			D	
conf-loc	element			D	
conf-name	element			D	
conf-num	element			D	
conf-sponsor	element			D	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
conf-theme	element			D	
conference	element			E	
content-type	attribute				
continued-from	attribute				IDREF
contrib	element			E	
contrib-group	element			E	
contrib-id	element			D	
contrib-id-type	attribute				
contrib-type	attribute				
copyright-holder	element			D	
copyright-statement	element			D	
copyright-year	element			D	
corresp	element			D	
corresp	attribute				
count	element			D	
count	attribute				
count-type	attribute				
country	element			D	
country	attribute				
counts	element			E	
currency	attribute				
custom-meta	element			E	
custom-meta-group	element			E	
data-title	element			D	
date	element			E	
date-in-citation	element			D	
date-type	attribute				
day	element			D	
deceased	attribute				
def	element			E	
def-head	element			D	
def-item	element			E	
def-list	element			E	
degrees	element			D	
description	attribute				
designator	attribute				
disp-formula	element			D	
disp-formula-group	element			E	
disp-level	attribute				
disp-quote	element			E	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
document-id	attribute				
document-id-type	attribute				
document-type	attribute				
dtd-version	attribute				
edition	element			D	
element-citation	element			E	
elocation-id	element			D	
elocation-id	attribute				
email	element			D	
end_date	attribute				
equal-contrib	attribute				
equation-count	element			D	
era	element			D	
etal	element			D	
executable	attribute				
ext-link	element			D	
ext-link-type	attribute				
fax	element			D	
fig	element			E	
fig-count	element			D	
fig-group	element			E	
fig-type	attribute				
fixed-case	element			D	
floats-group	element			E	
fn	element			E	
fn-group	element			E	
fn-type	attribute				
fontchar	attribute				
fontname	attribute				
format	attribute				
fpage	element			D	
frame	attribute				
front	element			E	
front-stub	element			E	
funding-group	element			E	
funding-source	element			D	
funding-statement	element			D	
given-names	element			D	
glossary	element			E	
glyph-data	element			P	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
glyph-data	attribute				IDREF
glyph-ref	element			D	
gov	element			D	
graphic	element			E	
headers	attribute				IDREF
history	element			D	
hr	element			D	
id	attribute				ID
initials	attribute				
inline-formula	element			D	
inline-graphic	element			E	
inline-supplementary-material	element			D	
institution	element			D	
institution-id	element			D	
institution-id-type	attribute				
institution-wrap	element			E	
isbn	element			D	
iso-8601-date	attribute				
issn	element			D	
issn-l	element			D	
issue	element			D	
issue	attribute				
issue-id	element			D	
issue-part	element			D	
issue-sponsor	element			D	
issue-title	element			D	
italic	element			D	
journal-id	element			D	
journal-id	attribute				
journal-id-type	attribute				
journal-meta	element			E	
journal-subtitle	element			D	
journal-title	element			D	
journal-title-group	element			E	
kwd	element			D	
kwd-group	element			E	
kwd-group-type	attribute				
label	element			D	
language	attribute				
language-version	attribute				

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
license	element			E	
license-p	element			D	
license-type	attribute				
link-type	attribute				
list	element			E	
list-content	attribute				
list-item	element			E	
list-type	attribute				
long-desc	element			D	
lpage	element			D	
media	element			E	
meta-name	element			D	
meta-value	element			D	
milestone-end	element			D	
milestone-start	element			D	
mime-subtype	attribute				
mimetype	attribute				
mixed-citation	element			D	
mml:math	element			E	
monospace	element			D	
month	element			D	
name	element			E	
name	attribute				
name-alternatives	element	X		E	
name-style	attribute				
named-content	element			D	
nested-kwd	element			E	
nlm-citation	element			E	
notation	attribute				
note	element			E	
notes	element		X	E	
notes-type	attribute				
object-id	element			D	
object-id	attribute				
object-id-type	attribute				
object-type	attribute				
on-behalf-of	element			D	
open-access	element			E	
orientation	attribute				
overline	element			D	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
overline-end	element			D	
overline-start	element			D	
p	element			D	
page	attribute				
page-count	element			D	
page-range	element			D	
part-title	element			D	
patent	element			D	
permissions	element			E	
person-group	element			D	
person-group-type	attribute				
phone	element			D	
platforms	attribute				
position	attribute				
postal-code	element			D	
prefix	element			D	
prefix-word	attribute				
preformat	element			P	
preformat-type	attribute				
price	element			D	
principal-award-recipient	element			D	
principal-investigator	element			D	
private-char	element			E	
product	element			D	
product-type	attribute				
pub-date	element			E	
pub-id	element			D	
pub-id-type	attribute				
pub-type	attribute				
publication-format	attribute				
publication-type	attribute				
publisher	element			E	
publisher-loc	element			D	
publisher-name	element			D	
publisher-type	attribute				
rationale	attribute				
rb	element			D	
ref	element			E	
ref-count	element			D	
ref-list	element			E	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
ref-type	attribute				
related-article	element			D	
related-article-type	attribute				
related-object	element			D	
resolution	attribute				
response	element			E	
response-type	attribute				
rid	attribute				IDREF
role	element			D	
roman	element			D	
rowspan	attribute				
rp	element			D	
rt	element			D	
ruby	element			E	
rules	attribute				
sans-serif	element			D	
sc	element			D	
scope	attribute				
season	element			D	
sec	element		X	E	
sec-meta	element			E	
sec-type	attribute				
self-uri	element			D	
seq	attribute				
series	element			D	
series-text	element			D	
series-title	element			D	
sig	element			D	
sig-block	element			D	
size	element			D	
source	element			D	
source-id	attribute				
source-id-type	attribute				
source-type	attribute				
span	attribute				
speaker	element			D	
specific-use	attribute				
speech	element			E	
start_date	attribute				
state	element			D	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
statement	element		X	E	
std	element			D	
std-organization	element			D	
strike	element			D	
string-conf	element			D	
string-date	element			D	
string-name	element			D	
style	attribute				
style-type	attribute				
styled-content	element			D	
sub	element			D	
sub-article	element			E	
subj-group	element			E	
subj-group-type	attribute				
subject	element			D	
subtitle	element			D	
suffix	element			D	
summary	attribute				
sup	element			D	
supplement	element			D	
supplement-type	attribute				
supplementary-material	element			E	
surname	element			D	
symbol	attribute				
table	element			E	
table-count	element			D	
table-wrap	element			E	
table-wrap-foot	element			E	
table-wrap-group	element			E	
target	element			D	
target-type	attribute				
tbody	element			E	
td	element			D	
term	element			D	
term-head	element			D	
tex-math	element			D	
textual-form	element			D	
tfoot	element			E	
th	element			D	
thead	element			E	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
time-stamp (deprecated)	element			D	
title	element			D	
title-group	element			E	
toggle	attribute				
tr	element			E	
trans-abstract	element		X	E	
trans-source	element			D	
trans-subtitle	element			D	
trans-title	element			D	
trans-title-group	element			E	
underline	element			D	
underline-end	element			D	
underline-start	element			D	
underline-style	attribute				
units	attribute				
unstructured-kwd-group	element			D	
uri	element			D	
valign	attribute				
verse-group	element			E	
verse-line	element			D	
version	element			D	
version	attribute				
vol	attribute				
volume	element			D	
volume-id	element			D	
volume-issue-group	element			E	
volume-series	element			D	
width	attribute				
word-count	element			D	
x	element			P	
x-size	attribute				
xlink:actuate	attribute				
xlink:href	attribute				
xlink:role	attribute				
xlink:show	attribute				
xlink:title	attribute				
xlink:type	attribute				
xml:base	attribute				
xml:lang	attribute				
xml:space	attribute				

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"

JATS Structure Name	Element or Attribute Property	Alternatives* Property	Section-like* Property	Whitespace Handling‡ Property	Attribute ID or IDREF Property
xmlns:ali	attribute				
xmlns:mml	attribute				
xmlns:xlink	attribute				
xmlns:xsi	attribute				
xref	element			D	
y-size	attribute				
year	element			D	

* In these columns, "X" means "yes", and no value means "no"

‡ In this column, "E" means "Element-like whitespace", "D" means "Data-like whitespace", and "P" means "Preserve whitespace"