Contents

Prepared by: Government Publishing Office

Published: September 30, 2025

1 Introduction

This Review Guide is intended to help users to understand changes in the 2.1 version of the United States Legislative Markup (USLM) schema so that users can provide meaningful feedback about the changes. This guide assumes that the reader is familiar with the 1.0 and 2.0, 2.0.12, 2.0.17, and 2.1 versions of the USLM schema and is generally knowledgeable about XML schemas in XSD format. For more information about previous versions, see the Existing Documentation section of this document for links to existing documentation.

This guide reflects USLM schema version 2.1.1. It is a draft version and will be updated in the near future.

2 Conventions Used in the User Guide

The following conventions are used in the User Guide:

- XML element names are denoted with angled brackets. For example, `<title>` is an XML element.
- Groups of elements are denoted with a word followed by the word 'elements'. For example, 'positioned note elements' includes any of the elements `<footnote>`, `<sidenote>`, `<endnote>`, and `<ear>`.
- XML attribute names are denoted with an "`@`" prefix. For example, `@href`. is an XML attribute.
- Enumerated values are denoted in a fixed width font. For example, `landscape` is an enumeration.
- String values are denoted with double quotes. For example, "`title1-s1`" is a string value.
- A new ***term*** being defined is shown in bold italic.
- A new **element** or **attribute** being defined is shown in bold.

3 Brief USLM Background

The USLM schema was first developed in 2013 by the Office of the Law Revision Counsel of the U.S. House of Representatives (OLRC) in order to produce the United States Code in XML. Since 2013, the OLRC regularly produces a USLM version of the United States Code for download at http://uscode.house.gov/download/download.shtml. The USLM version of the U.S. Code is updated continuously as new laws are enacted.

The original goals of the USLM schema included:

1. *Allow existing titles of the United States Code to be converted into XML.*
2. *Support ongoing maintenance of the United States Code.*
3. *Support the drafting of new positive law codification bills and related materials.*
4. *Provide a flexible foundation to meet future needs of Congress.*
5. *Compatibility with existing legislative documents in other XML formats.*

Building on the "flexible foundation" in goal number four above, the Government Publishing Office (GPO) released the 2.0 update to USLM that extended its use to the following document sets:

- Enrolled Bills and Resolutions
- Public and Private Laws
- Statutes at Large
- Statute Compilations
- Federal Register (FR)
- Code of Federal Regulations (CFR)

The documentation for the USLM used in these document sets is in the 2.0.12 version of the Review Guide.

Further additions were made to USLM 2.0 to enable its use for bills and resolutions in all document stages. This is documented in the 2.0.12 and 2.0.17 versions of the Review Guide.

The changes made to the USLM schema to support its use for amendment documents were breaking changes and the schema numbering 2.1 reflected that. This is documented in the 2.1 version of the Review Guide.

4 Existing Documentation

User documentation for the 1.0 version of the schema can be found at https://github.com/usgpo/uslm/blob/main/USLM-User-Guide.pdf and https://github.com/usgpo/uslm/blob/main/USLM-User-Guide.md.

User documentation for the 2.0 version of the schema, to version 2.0.12, can be found at https://github.com/usgpo/uslm/blob/main/USLM-2_0-Review-Guide-v2_0_12.md and https://github.com/usgpo/uslm/blob/main/USLM-2_0-Review-Guide-v2_0_12.pdf.

User documentation for the 2.0 version of the schema after 2.0.12, to version 2.0.17, can be found at https://github.com/usgpo/uslm/blob/main/USLM-2_0-Review-Guide-v2_0_17.md and https://github.com/usgpo/uslm/blob/main/USLM-2_0-Review-Guide-v2_0_17.pdf.

User documentation for the 2.1.0 version of the schema can be found at https://github.com/usgpo/uslm/blob/main/USLM-2_1-ReviewGuide.md and https://github.com/usgpo/uslm/blob/main/USLM-2_1-ReviewGuide.pdf.

The XSD schema and CSS stylesheets for online viewing can be downloaded from https://github.com/usgpo/uslm. Note that the CSS stylesheet is informational only. It produces a draft view of the documents.

Note: These resources and more are available on GPO's Developers Hub at https://www.govinfo.gov/developers.

5 What Has Not Changed

Version 2.1.1 of USLM is architecturally an incremental change to the schema. While many new elements have been added and several content models have been extended or modified, the fundamental design of the schema has not changed. The following principles, documented in the 1.0 and 2.0 versions of the User Guide, continue in version 2.1.1:

- Abstract and Concrete Models
- Inheritance
- Attribute Model
- Core Document Model
- Metadata Model

- Hierarchy Model
- Versioning Model
- Presentation Model
- Relationship to HTML
- Identification Model
- Referencing Model

Many of these models have been extended to accommodate the additional document types and their structures. These extensions are backwards-compatible except in a few cases described below.

6 Schema Changes

6.1 New Document Type: Legislative Reports

Legislative reports are the reports created by committees to accompany legislation. As such, they are not legislative documents although they may contain legislative content. They have different metadata requirements and a much looser content structure than legislative documents. For this reason USLM 2.1.1 contains a new document type, **<legislativeReport>**. This document type uses new elements that are equivalent to the existing <meta>, <preface>, and <main> elements, namely **<reportMeta>**, **<reportPreface>**, and **<reportMain>**. As in other legislative documents, the <reportMeta> element contains the machine-processable metadata, <reportPreface> the metadata that is designed to be shown with the rest of the document content, and <reportMain> contains the main content of the <legislativeReport> document.

In the future we expect to model different types of reports.

6.2 New Elements

Many elements may occur in more than one location in a report; they are described in the context of the first element in which they occur, not all elements.

6.2.1 Legislative Report

The new legislative report element is **<legislativeReport>**. It is the type of committee report that accompanies proposed legislation.

6.2.2 Report Meta

The **<reportMeta>** element contains machine-processable metadata for the entire report. The new elements allowed as child elements are documented below.

<accompanies> The <accompanies> element contains citable information about the proposed bill or resolution that the report concerns. In the <reportMeta> element the format is the same as the <citableAs> format for the bill or resolution; in the <reportPreface> the format is usually one of

```
<accompanies>[To accompany <affected>
    <dc:type>...</dc:type> <docNumber>...</docNumber>
</affected>]</accompanies>
```

or

```
<accompanies>on <affected>
    <dc:type>...</dc:type> <docNumber>...</docNumber>
</affected></accompanies>
```

<cboCostEstimateLine> The <cboCostEstimateLine> notifies the reader if a report contains a CBO cost estimate. In the <reportMeta> this has the values 'Y' (yes) or 'W' (waived). In the <reportPreface> it contains a string with the relevant information. If there is no value, or the element is not present, there is no information as to whether the cost estimate was waived or not. The attribute @value is added in the <reportPreface>, with the values 'Y' (yes) or 'W' (waived).

<genre> The <genre> contains a category or type and is used in reports. If a document fits into multiple genres, one element per genre should be used. An example is "legislative" for legislative committee reports, or both "legislative" and "conference" for legislative conference committee reports.

<jacketId> The <jacketId> is a publishing number that is sometimes printed on the report.

6.2.3 Report Preface

<reportCoverPage> The <reportCoverPage> contains the cover page of a committee report and may also contain a spine for a bound report. It is often repeated in the report as a report jacket.

<reportTitle> The <reportTitle> contains the title of a report.

<reportViews> The <reportViews> notifies the reader if the document contains supplemental views, such as minority or additional views.

<legislator> The <legislator> element contains the name for each legislator who contributed to the legislative report, or whose vote is recorded. There are a number of optional attributes.

The **@bioGuideId** attribute is the BioGuide ID used for the Member (House or Senate) of Congress.

The **@committeeRole** attribute is the role the Member (House or Senate) of Congress has for the committee in this report.

The **@partyAffiliation** attribute is the party affiliation the Member (House or Senate) of Congress as of the date this report is published.

The **@caucusAffiliation** attribute is the party that the Member (House or Senate) of Congress caucuses with as of the date this report is published.

The **@rank** attribute is the relative rank that the Member (House or Senate) of Congress has within the party or caucus for this report.

<legislators> The element contains the list of legislators who contributed to the legislative report.

<committeeMembersAndStaff> The <committeeMembersAndStaff> is a listing of legislators and staff that is used in legislative and committee reports.

<staff> The <staff> element contains the name and optionally other information about a person who is not a legislator who contributed to the legislative report.

<staffList> The <staffList> element contains the list of staff who contributed to the legislative report. This includes all staff considered to have contributed, whether full-time employees, contractors, or others.

6.2.4 Report Main

The **<reportMain>** element consists of a number of units of content, modeled as **** elements. Most segments in a committee report have a simple structure using generic elements already in USLM 2, such as <heading>, <p>, and <list>. There are specialised elements for those segments for which the simple model does not apply. This includes segments for legislative content (**<legislativeSegment>**), changes in existing law to fulfill the Ramseyer and Cordon rules (**<changesInExistingLaw>**), committee votes

(<votesInCommittee>), and supplemental views such as minority or additional views (<supplementalViews>). More segments will be defined in future, for example for the CBO Cost Estimate.

6.2.5 Recommendation

Most legislative reports include a <recommendation> from the committee as to whether the legislation should be passed, passed with amendments, or otherwise. The <recommendation> element, if present, is first in the document, or occurs after the Table of Contents. Usually the Table of Contents, if present, occurs after the <recommendation>.

6.2.6 Segment

The is the principal generic unit of content in a committee report. It is used for all non-specialised units of content, and contains existing USLM 2 elements such as <p>, <heading>, <subheading>, <num>, <list>, <figure>, and <xhtml:table>. The @role attribute can be used to indicate a specific type of segment, such as "purpose".

Even where the content seems to indicate hierarchical levels, the element does not use hierarchies. <heading> elements are used either with appropriate classes, or by using a @role attribute together with the appropriate @styleType attribute to effect the desired formatting.

6.2.7 Legislative Segment

The <legislativeSegment> is typically the first segment in a legislative report, occurring after the <recommendation> and the optional <toc>. It, unlike most segments in a committee report, contains legislative content, typically in the form of an amendment. A common pattern is an amendment in the nature of a substitute. Example:

```
<legislativeSegment role="amendment"><p>The amendment is as
   follows:</p>
  <amendmentInstruction><content>Strike all after the enacting
     clause and insert the
following:
    <amendmentContent><section><num>SECTION 1. </num>
      <heading>SHORT TITLE.</heading>
      ...
    </amendmentContent></content>
  </amendmentInstruction></legislativeSegment>
```

The <legislativeSegment> element may also be used in a supplemental view.

6.2.8 Changes in Existing Law Segment

The **<changesInExistingLaw>** segment meets the requirements of the Cordon and Ramseyer constraints and contains the changes to existing law that would be made by the proposed legislation. The structure of this segment allows for the usual segment content as well as specific content containing the changes. The elements used for this are

<existingLaws> Contains the collection of existing laws which the proposed legislation would change.

<existingLaw> A portion of an existing law which the proposed legislation would change. This may have an attribute **@source** with a citable version of the existing law's identifier. Examples are
source="/us/scomp/115-23@118-34" to indicate a statute compilation for Public Law 115-23 that is valid through public law 118-34.
source="/us/usc/t42@118-34" to indicate Title 42 in the US Code that has release point public law 118-34.
source="/us/bill/114/hjres/34" to indicate the enrolled bill H.J. Res 114-34.
source="/us/pl/115/23" to indicate the Public Law 115-23 (not the Statute Compilation for it).

<existingLawName> The name of the existing law which the proposed legislation would change. This includes which section or title of the existing law would be changed if the changes are limited to a section or title. This element is a child element of a <heading> or <subheading> element.

<proposedContent> The changes to be made are shown as deleted text (using the already-existing <deletedText> element), added text (using the already-existing <addedText> element), or a mixture of text and hierarchy, for which the new <proposedContent> element is used. This last must contain an attribute @changed which details whether the content within is added or deleted.

Example:

```
<changesInExistingLaw>
 <heading class="smallCaps">Changes in Existing Law Made by the
    Bill, as Reported</heading>
 <p>...</p>
 <existingLaws>
  <existingLaw source="/us/pl/93/531">
    <heading>Public Law 93-531</heading>
```

```
    <subheading><existingLawName>AN ACT Relating to the
       ...</existingLawName></subheading>
    <changesToLaw>
    content can be multiple provisions with minimal context or an
       entire document
...</changesToLaw>
  </existingLaw>
<!-- Repeat for each law that is changed -->
 </existingLaws>
</changesInExistingLaw>
```

6.2.9 Estimated Costs Segment

The **<estimatedCosts>** segment element contains the estimated costs for the associated legislation. This segment allows the usual segment content, as well as elements for the estimate from the Congressional Budget Office (CBO), if one is included in the report.

The elements for the CBO estimate are **<cboLetter>** for the letter, and **<cboCostEstimate>** for the estimate. The content models for these two elements are still under development and will be added to a future version of USLM 2.1.

6.2.10 Votes in Committee Segment

The **<votesInCommittee>** segment element contains the information about votes that were taken in the committee about the proposed legislation. It contains the usual segment content, with the addition of zero or more tally sheet elements that are used to record the votes.

6.2.11 Tally Sheet

The **<tallySheet>** element is used to record legislator votes for one motion. Each <tallySheet> can be part of a legislative report, or a stand-alone document with its own metadata. When the tally sheet is incorporated into the report, the metadata (**<tallySheetMeta>** and **<tallySheetPreface>** elements) can be retained or discarded, along with the recorded votes. Alternatively the data of the recorded votes can be retained, with the display turned off.

6.2.11.1 Tally Sheet Meta The <tallySheetMeta> element contains new elements. Most of these elements may also occur in the body of the tally sheet.

<event> The <event> element contains information about the meeting, in its child elements.

<eventDate> The <eventDate> is the date of the meeting event at which the vote occurred.

<eventId> The <eventid> is an identification string for the event, limited to 128 characters.

<eventTitle> The <eventTitle> is the title of the event or meeting at which the vote occurred.

<majorityParty> The majority party as of the date of the vote.

<motionOfferedBy> The legislator who offered (moved) the motion that is the subject of the vote.

<tallySheetId> The <tallySheetId> is an identification string for a tally sheet used to record votes that is system-generated. It is limited to 128 characters, and it is not the same as the vote or roll call number.

<voteDate> The <voteDate> is the date of the vote in a tally sheet. This is not necessarily the same as the meeting event date.

<modVoteDate> A <modVoteDate> is the date of the re-considered vote in a tally sheet under the Senate process, whereby a Senator may change their vote within a specified time period.

<updateVoteDate> The <updateVoteDate> is the date clerical updates (if any) were made to a vote in a tally sheet. This is allowed under the House process for clerical edits that do not change the result.

<voteDescription> The <voteDescription> describes the vote that is the subject of the tally sheet, with information to specify the amendment (if applicable), the amendment sponsor <sponsor> (if applicable), the motion sponsor <motionOfferedBy> (if applicable), and the purpose of the vote. If the <voteDescription> includes a number, this number is not necessarily the same as the roll call <voteNum> number.

<voteDisposition> The <voteDisposition> contains text with the disposition of the vote, e.g., 'failed', 'rejected', or 'passed'.

<voteNum> The <voteNum> is the roll call number or identifying number for one committee vote or tally sheet, not the vote of one person. (The vote of one person has the element name <vote>.) The content of the <voteNum> is text, potentially with formatting when used in the tally sheet preface. The formatting is determined and chosen by the committee. It is not necessarily the same number as that used (if any) in the <voteDescription>.

<voteType> The <voteType> element describes the voting method used in the vote that is recorded in the tally sheet. Examples are roll call, voice vote, unanimous, or withdrawn.

<voteTotals> The <voteTotals> element contains the elements with the numbers of votes

of a particular type for a single tally sheet. It can also contain elements used for formatting, when it is used in the rendered portion of the tally sheet. The child elements it contains are
<absentTotal>: number of votes of 'absent'
<ayeVoteTotal>: number of votes of 'aye'
<nayVoteTotal>: number of votes of 'nay'
<noVoteTotal>: number of votes of 'no'
<notVotingTotal>: number of votes of 'not voting'
<otherVoteTotal>: number of votes of some other value
<presentVoteTotal>: number of votes of 'present'
<yeaVoteTotal>: number of votes of 'yea'

Required Elements

The elements <congress>, <currentChamber>, <voteNum>, and <voteType> are required in the <tallySheetMeta> element.

6.2.11.2 Tally Sheet Preface The **<tallySheetPreface>** element can contain the same elements as the <tallySheetMeta>, with the addition of formatting elements such as <p> and <heading>.

6.2.11.3 Recorded Votes The **<recordedVotes>** element contains a collection of **<recordedVote>** elements. Each <recordedVote> element contains one **<vote>** element with the textual value of the vote ("aye", "nay", etc) and one <legislator> element with the information about the legislator who cast the vote. Display of this part of the tally sheet can be turned off with the @display attribute set to no.

The attribute **@byProxy** on the is <recordedVote> element is used to show whether a legislator's vote was by proxy ('yes') or not ('no'). The default value is 'no', the vote was not by proxy.

6.2.12 Supplemental Views

The **<supplementalViews>** element contains the supplemental (such as minority or additional) views that are published in a committee report. It can include and <legislativeSegment> elements as well as the usual segment content.

6.3 Conference Reports

Although Conference Reports are agreed on by both Chambers, they are published by one Chamber (typically the House, although on rare occasions the Senate will also publish a Conference Report). The <currentChamber> element in the <reportMeta> declares the Chamber that published the report.

Conference Reports (and Supplemental Views) are signed by multiple sets of legislators. For the purpose of grouping multiple sets of <signatures> elements, there is a new **<signaturesBlock>** element.

6.4 Deleted Elements or Attributes

The @temporalId attribute is not used in USLM. Accordingly, it has been removed. There are other mechanisms for identifying time-based versions of a document, which remain.

6.5 Added Guidance for Usage

6.5.1 Document Identification

The combination of <dc:type>, <docNumber>, and <congress> suffices to identify a document in most document types. The exception is Conference Reports, where the <currentChamber> information must be added to provide sufficient information to locate the document in Government systems such as GovInfo, Congress.gov, or Chamber document repositories. The <citableAs> elements contain the needed information.

In Conference Reports the <relatedDocuments> element should be used to identify the precise version of each bill, resolution, or amendment document that was the subject of the conference.

6.5.2 Sluglines

The <slugLine> is a line added to the <reportPreface> (and other types of preface elements) that will be printed in the page footer. The author does not need to be concerned about the placement of the slugLine in the preface as the composition engine will place the content in the appropriate location on the rendered page. By convention it is the last child of the appropriate preface (in this case the <reportPreface>) element.

7 Schematron

7.1 Introduction

As use of the USLM schemas increases, different stakeholder groups with different objectives and requirements will use USLM to represent and process content. This introduces situations where different content rules are desirable. For example, markup that is acceptable or even necessary and desirable in a pre-introduced bill might not be acceptable in a passed bill. Representing this variation in the schema would introduce substantial overhead from complexity.

The stakeholders have agreed that the more appropriate approach going forward is to maintain the 2.* USLM schemas with only those restrictions which are applicable to all USLM users. Restrictions specific to a particular document set or to a processing mechanism will need to be implemented by mechanisms other than the official USLM XSD schemas.

One such mechanism is the use of a secondary validation mechanism that uses the ISO Standard language Schematron. This is a widely-accepted secondary validation language that uses XPath to describe constraints that the document instance must follow, that depend on the document type. These constraints, once expressed in XPath, can be expressed in other programming languages for other systems and thus a Schematron schema can be used either directly or as a source of information on the applicable constraints for a given document type. Additional constraints can be added to support processes such as drafting and editing bills and resolutions, or drafting legislative reports.

7.2 Using Schematron

The Schematron is not designed to replicate the validation available from the XSD schema. The expectation is that validation is a two-step process. A well-formed document of a given document type should be validated against the USLM 2.1.1 schema, and can then be further validated against the appropriate Schematron file for that document type.

Although use of the Schematron for additional validation is not required, downstream tools that rely on the documents meeting the constraints that are tested in the Schematron may not process documents correctly that fail Schematron validation for that document type and use case. We envisage different tools using different versions of Schematron validation to match their use case. As an example, the Schematron used for a drafting tool may check that the documents pass one set of constraints, while a typesetting tool may require that documents satisfy a different set of constraints, both sets of constraints being expressed

in different Schematron schemas, and all these versions of the documents being validated against the USLM schema.

Although a Schematron schema is not included in the current release of USLM 2, one is in development. The first release will be a Schematron for validating legislative report documents.

8 Feedback

To submit feedback, questions, or comments about the USLM 2.1.1 schema and this Review Guide, please open a GitHub issue at https://github.com/usgpo/uslm/issues.