Quality Control (QC) Vocabulary XML

CONTENTS

NOTICES

REVISION HISTORY

Version	Date	Description
1.0	May 20, 2019	First Release

1 INTRODUCTION

QC Vocabulary defines terms that are used in QC with the goals of 1) Ensuring that terms are used consistently, and 2) ensuring that terms are encoded consistently.

his document defines the structure for encoding information about Quality Control (QC) vocabulary. In particular, it describes the precise meaning of QC terms, including error terms, and the proper and consistent way to encode them. Companion to this document are HTML and XML that include these ratings data.

These were created with the goal of supporting interoperability between those generating content, and those performing QC evaluation of that content.

1.1 Scope

The QC Vocabulary itself is captured in XML and HTML at www.movielabs.com/md/qcvocabulary.

This document describes

- How to interpret the QC Vocabulary found in the XML and HTML
- The schema defining the structure for the QC Vocabulary XML

1.2 Relationship to Other Specifications

QC Vocabulary can be used wherever it is necessary to exchange information in the context of QC. However, it was developed in the context of content delivery within the context of the MovieLabs Digital Distribution Framework (MDDF), found at www.movielabs.com/md.

MDDF specifications expect the QC Vocabulary to be used when QC data is exchanged.

1.3 Document Organization

This document is organized as follows:

1. Introduction—background, scope and conventions
2. Interpreting QC Vocabulary
3. QC Vocabulary Schema

1.4 Document Notation and Conventions

As a general guideline, the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]. That is:

- “MUST”, “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification.

- “MUST NOT” or “SHALL NOT” means that the definition is an absolute prohibition of the specification.
- “SHOULD” or “RECOMMENDED” mean that there may be valid reasons to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- “SHOULD NOT” or “NOT RECOMMENDED” mean that there may be valid reasons when the particular behavior is acceptable, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- “MAY” or “OPTIONAL” mean the item is truly optional, however a preferred implementation may be specified for OPTIONAL features to improve interoperability.

Terms defined to have a specific meaning within this specification will be capitalized, e.g. “Track”, and should be interpreted with their general meaning if not capitalized.

Normative key words are written in all caps, e.g. “SHALL”.

Normative requirements need not use the formal language above.

1.4.1 XML Conventions

XML is used extensively in this document to describe data. It does not necessarily imply that actual data exchanged will be in XML. For example, JSON may be used equivalently.

This document uses tables to define XML structure. These tables may combine multiple elements and attributes in a single table. Although this does not align with schema structure, it is much more readable and hence easier to review and to implement.

Although the tables are less exact than XSD, the tables should not conflict with the schema. Such contradictions should be noted as errors and corrected.

1.4.1.1 Naming Conventions

This section describes naming conventions for Common Metadata XML attributes, element and other named entities. The conventions are as follows:

- Names use initial caps, as in InitialCaps.
- Elements begin with a capital letter, as in InitialCapitalElement.
- Attributes begin with a lowercase letter, as in initiaLowercaseAttribute.
- XML structures are formatted as Courier New, such as `md:id-type`
- Names of both simple and complex types are followed with “-type”

1.4.1.2 Structure of Element Table

Each section begins with an information introduction. For example, “The Bin Element describes the unique case information assigned to the notice.”

This is followed by a table with the following structure.

The headings are

- Element—the name of the element.
- Attribute—the name of the attribute
- Definition—a descriptive definition. The definition may define conditions of usage or other constraints.
- Value—the format of the attribute or element. Value may be an XML type (e.g., “string”) or a reference to another element description (e.g., “See Bar Element”). Annotations for limits or enumerations may be included (e.g.,” int [0..100]” to indicate an XML xs:int type with an accepted range from 1 to 100 inclusively)
- Card—cardinality of the element. If blank, then it is 1. Other typical values are 0..1 (optional), 1..n and 0..n.

The first row of the table after the header is the element being defined. This is immediately followed by attributes of this element, if any. Subsequent rows are child elements and their attributes. All child elements (i.e., those that are direct descendants) are included in the table. Simple child elements may be fully defined here (e.g., “Title”, “ ”, “Title of work”, “xs:string”), or described fully elsewhere (“POC”, “ ”, “Person to contact in case there is a problem”, “md:ContactInfo-type”). In this example, if POC was to be defined by a complex type defined as md:ContactInfo-type. Attributes immediately follow the containing element.

Accompanying the table is as much normative explanation as appropriate to fully define the element, and potentially examples for clarity. Examples and other informative descriptive text may follow. XML examples are included toward the end of the document and the referenced web sites.

1.4.2 General Notes

All required elements and attributes must be included.

When enumerations are provided in the form ‘enumeration’, the quotation marks (‘’) should not be included.

UTF-8 [RFC3629] encoding shall be used when ISO/IEC 10646 (Universal Character Set) encoding is required.

1.5 Normative References

[TR-META-CR] *Common Metadata Content Ratings*, TR-META-CR, www.movielabs.com/md/ratings. Note that a specific version is not referenced as it is intended that the latest version will be used. Referencing specifications may selection a specific version of the referenced document.

[RFC2141] R. Moats, *RFC 2141*, *URN Syntax*, May 1997, http://www.ietf.org/rfc/rfc2141.txt

[RFC3986] Berners-Lee, T., et al, RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC5646] Philips, A, et al, *RFC 5646, Tags for Identifying Languages*, IETF, September, 2009. http://www.ietf.org/rfc/rfc5646.txt

[XML] "XML Schema Part 1: Structures", Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-1/ and "XML Schema Part 2: Datatypes", Paul Biron and Ashok Malhotra, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/

1.6 Informative References

None

1.7 Best Practices for Maximum Compatibility

Metadata typically evolves with the addition of new elements, attributes and vocabularies. Existing applications should be capable of accepting metadata, even though there might be more data than expected. Strict XML validation precludes an orderly evolution and can be counterproductive to the flexibility needed in real implementations.

Metadata specifications and schema updates are designed to support backwards compatibility. For example, element and attributes can be added, but required elements are not removed; or more generally ordinality of elements and attributes can be widened but not narrowed. Values are not changed in either syntax or semantics. Therefore, we strongly encourage implementations to either be diligent in tracking to the latest version, or follow the backwards compatibility rules provided here.

An XML document is considered compatible if its structure does not preclude the extraction of data from the document. For example, a document with additional elements and attributes do not preclude schema parsing and data extraction.

- Do not reject compatible XML documents, unless they fail schema validation against the definition for an exact version/namespace match.
- Extract data from compatible XML documents whenever possible
- It is allowable to ignore elements and attributes whose presence is not allowed in the specification and schema versions against which the implementation was built. For example, if the original schema allows one instance and three instances are found, the 2nd and 3rd instance may be ignored.

We will try to update metadata definitions such that following these rules work consistently over time. Sometimes, changes must be made that are not always backwards compatible, so we will do our best to note these.

2 INTERPRETING QC VOCABULARY

QC Vocabulary might grow to cover is primarily focused on error terms, although some additional terms are included.

2.1 Categories and Terms

The QC Vocabulary list is grouped into different categories such as "Audio" and "Video" to make the list more manageable. Within each category are tems.

The vocabulary item itself is the combination of <category> and <term>; for example, "Video Aliasing". "Aliasing on its own is meaningless. This is important because some terms are used in more than one category; for example, "Video Other" and "Audio Other".

The list of categories is maintained at www.movielabs.com/md/qcvocabulary. We do not include the list here as it changes over time and is not substantive to this document.

2.2 QC Vocabulary Data

The QC Vocabulary data, whether it is in XML or HTML, includes these four parts:

- Category – Indicates the type of resource (i.e., deliverable) an error is applicable to, such as video, audio, artwork, etc. The Category may, therefore, be used within automated workflows to route error reports to the appropriate team for handling.
- Term – A term associated with a particular error. A term may be associated with more than one Category. The combination of Category and Term, however, will uniquely identify an error.
- Description – A short description of the error
- Definition – The formal definition of the error, to distinguish it from all other errors.

Category and Term are defined to be computer-readable for use in automated message processing.

There is additional data for bookkeeping

- Version – the version of the release that included this term.
- Date – The date the term was last updated
- Deprecated – Indicates the term is no longer in use and, at some point, will fail validation. It is included for reference to avoid confusion.
- Related terms – if there are other terms that are similar, they are include here. Generally, the Definition will indicate how to differentiate. This is informational and not guaranteed to be complete.
- Link to Example – If an illustrative examples is available, a link may be included.

3 QC VOCABULARY SCHEMA

The QC Vocabulary Schema as designed to map to a QC Vocabulary Excel spreadsheet. Consequently, the structure is very flat and simple. Each entry in the XML maps to a single row in the spreadsheet.

3.1 QCVocabulary Element and QCVocab-type Complex Type

The root level element is QCVocabulary. It is defined by the complex type QCVocab-type.

QCVocab-type contain an entry for each QC Vocabulary term.

Element	**Attribute**	**Definition**	**Value**	**Card.**
QCVocab-type				
Entry		QC Vocabulary Term	QCVocabEntry-type	0..n

3.1.1 QCVocabEntry-type Complext Type

The QCVocabEntry-type contains a single term. It is defined as follows:

Element	**Attribute**	**Definition**	**Value**	**Card.**
QCVocabEntry-type				
Category		As described in Section 2.2	xs:string	
Term			xs:string	
Description			xs:string	
Definition			xs:string	
Version			xs:string	
Date			xs:string	
Deprecated			xs:string	
RelatedTerms			xs:string	
LinkToExample				

4 SCHEMA

This section includes the schema and how it is used with Excel.

4.1 Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" elementFormDefault="qualified"
attributeFormDefault="unqualified" vc:minVersion="1.1">
   <!--QC Vocabulary-->
   <!--Verson 1.0 DRAFT-->
   <!-- *** QC Vocabulary ***-->
   <xs:complexType name="QCVocabEntry-type">
      <xs:sequence>
         <xs:element name="Category" type="xs:string"/>
         <xs:element name="Term" type="xs:string"/>
         <xs:element name="Description" type="xs:string"/>
         <xs:element name="Definition" type="xs:string"/>
         <xs:element name="Version" type="xs:string"/>
         <xs:element name="Date" type="xs:string"/>
         <xs:element name="Deprecated" type="xs:string"/>
         <xs:element name="RelatedTerms" type="xs:string"/>
         <xs:element name="LinkToExample" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="QCVocab-type">
      <xs:sequence>
         <xs:element name="Entry" type="QCVocabEntry-type" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="QCVocabulary" type="QCVocab-type"/>
</xs:schema>
```

4.2 Using Schema with Excel (hints)

This section includes some information on using Excel's XML output feature. This was writing in May 2019, so some items may be different for readers in the future.

Here are a few steps you might find useful:

- Open spreadsheet.
- Select Developer tab. If you don't see a Developer tab, you will need to turn it on in Excel Options.
- Select Source. You should see an XML Source window open.
- If the XML Source window includes QCVocabulary/Entry/ with various entries in bold font, select Export. XML will be exported and you're done.
- If the XML Source Window is empty, select "XML Maps…". It will open a window.
- Select "Add…", provide the name of the file with the schema, and select Open. Note that you can do the same thing with an XML document in the right format, but you'll have to be careful to rewrite the whole file. Select OK.

- Back to XML Source, you need to map the leaf elements to columns. Select a cell in the column to map (in the spreadsheet), then double-click the name under Entry (in XML Source). You can also drag the name to the column in question.
- Once all the columns are mapped, select Export from the Developer tab.