WIPO ST.96 XSD Validator Manual

Version 1.1

June 1, 2020

Version History

Change date	ST96XSDValidator version	Description of changes
2016-06-01	1.0	Release based on WIPO Standard ST96 Annex I V2.1
2020-05-26	1.1	Release based on WIPO Standard ST96 Annex I V4.0

Table of Contents

Introduction

1.1   Purpose

This document describes ST96XSDValidator, a tool that uses Schematron to validate XML Schemas (XSDs) against *WIPO Standard ST.96, Annex I - IP XML Design Rules and Conventions (DRCs)*.

1.2   Benefits

Automated enforcement of *WIPO Standard ST.96, Annex I - IP XML Design Rules and Conventions:*

- promotes interoperability and sound data exchange by enforcing a standard set of rules and guidelines,

- ensures consistency throughout the intellectual property communities and helps to accommodate growth and change by enforcing flexible XML design principles,

- decreases the level of effort required for XML developers to follow the standards by providing automated enforcement for most rules and a clear checklist for rules that must be manually checked.

1.3   Scope

The scope of this document includes validation of conformance WIPO Standard ST.96 rules pertaining to the creation of XML Schemas. Rule pertaining to the creation of XML document instances are out of scope.

Validating Conformance to ST.96 DRCs

WIPO ST.96 XML Design Rules and Conventions defines a comprehensive set of constraints to be followed when creating XSDs related to patent, trademark, and design Industrial Property (IP) types. XSDs should be validated to determine whether they conform to these rules. Sometimes this validation process can be automated, but in other cases it requires manual assessment.

An automated tool can enforce rules that are expressed mechanically. For example, a rule requiring the length of component names to be less than 35 characters (GD-07) can be automated. Automation can enforce rules that state what is prohibited. For example, a rule stating that schemas must not use xsd:redefine (SD-07) can be automated.

An automated tool *cannot* enforce rules requiring human judgement. For example, a rule calling for names to be self-explanatory (GD-08) cannot be automated. Automatation also cannot enforce rules that state what is permitted rather than what is prohibited. For example, a rule stating that abstract types may be used (SD-47) involves nothing to check because it declares only that which is already allowed by W3C XML Schemas.

Appendix A has a table describing whether automation is possible and whether enabling assumptions are required each XSD rule in *WIPO ST.96 XML Design Rules and Conventions*.

About Schematron

Schematron is an ISO Standard (ISO/IEC 19757-3:2006) schema language that allows for the expression of constraints on XML documents. While most grammar-based schema languages (such as W3C XML Schema) can specify the general structure and valid values of an XML document, Schematron is much more flexible in terms of the kinds of constraints or "business rules" it can enforce. It uses XPath to express these constraints, and provides a custom vocabulary that helps to document the individual rules and the desired reporting procedures for violations of these rules.

The following code sample shows a Schematron pattern that enforces GD-07 (*the maximum length of a component name SHOULD be no more than 35 characters*):

```
<pattern>
    <title>GD-07</title> ①
    <rule context="②  xsd:complexType[@name]
                    | xsd:simpleType[@name]
                    | xsd:element[@name]
                    | xsd:attribute[@name]">
        <assert test="string-length(@name) &lt;= 35" ③
                flag="AUTO" ④
                role="WARNING" ⑤>
        ⑥  The length of the
            <value-of select="local-name(.)"/>
            named
            <value-of select="@name"/>
            is
            <value-of select="string-length(@name)"/>.
        </assert>
    </rule>
</pattern>
```

The major parts of this Schematron pattern work together to enforce the GD-07 rule:

① This pattern's **title** specifies the rule identifier, GD-07, implemented by this pattern.
② The **context** of this rule specifies that GD-07 applies to named components (complexTypes, simpleTypes, elements, and attributes) in the input XSD.
③ The **assertion test** specifies that GD-07 requires names to be less than 35 characters long.
④ The **assertion flag** indicates that this rule can be determined AUTOmatically.
⑤ The **assertion role** indicates that the user should be WARNed if this assertion's test proves not to be true. Violation of any rule using the SHOULD keyword results in a warning; violation of any rule using the MUST keyword results in an error.
⑥ The **assertion violation message** specifies the warning to display to when this assertion's test fails.

For a general introduction on ISO Schematron see *ISO Schematron tutorial: An introductory guide*.

Installing and Configuring ST96XSDValidator

1.4   Requirements

The only external dependency required to run ST96XSDValidator is Java JDK 1.5 or later. Everything else is included in the installation zip file.

1.5   Installation

1. [Install Java JDK](#) if it is not already installed on your machine. Java JRE is not sufficient. Java JDK is needed as it includes the needed tools.jar file. Note the location where Java JDK is installed.
2. Download ST96XSDValidator_*version*.zip.
3. Unzip ST96XSDValidator_*version*.zip into an installation directory.
4. Update ST96XSDValidator/dev/settings.xml file with the Java JDK installation path from step #1. For example:
    ```
    <JDKHome>"C:\Program Files\Java\jdk1.8.0_11"</JDKHome>
    ```
5. **[For non-Windows machines only]**   From the ST96XSDValidator/dev installation directory, type "source permissions" to give ST96XSDValidator, runtests and ant execution permissions.

The ST96XSDValidator is now ready to run. The remaining subsections in Section 0 describe optional configuration settings. You may skip ahead to Section 0 if you wish to run ST96XSDValidator in its default configuration.

1.6   IP Office Specific Configuration

This section describes *optional* settings for specifying IP Office specific configuration. It is not required in order to use ST96XSDValidator.

Most ST.96 rules apply equally well to ST.96 XSDs and IPO-specific XSDs. The validator may be run without change on IPO-specific XSDs, and most of the rules will be checked just fine. For example, GD-07 checks that the maximum length of component names does not exceed 35 characters. This check will apply equally well to IPO-specific XSDs.

However, a rule such as SD-03, which says that Patent schemas must not refer to Trademark schemas (and vice-versa), can also be made to work for IPO-specific XSDs by providing IPO-specific namespace configuration information in the file, ipo.xml.

A sample ipo.xml configured with the United States Patent and Trademark Office (USPTO) namespace information is provided. By default, the information is commented-out but can serve as a template for configuring ST96XSDValidator to consider IPO-specific XSDs. Simply copy the common, patent, design, or trademark elements out of the commented section and modify the attributes for any IPO-specific namespace information you wish to be considered during validation. This is entirely optional; validation does not require that any IPO-specific configuration be made.

Here is the default ipo.xml. USPTO settings are included as examples but are commented out:

```
<ipo>
  <!--
  <common subdir="USCommon"
          ns="urn:us:gov:doc:uspto:common"
          ns-prefix="uscom"/>
  <patent subdir="USPatent"
          ns="urn:us:gov:doc:uspto:patent"
          ns-prefix="uspat"/>
  <design subdir="USDesign"
          ns="urn:us:gov:doc:uspto:design"
          ns-prefix="usdgn"/>
  <trademark subdir="USTrademark"
          ns="urn:us:gov:doc:uspto:trademark"
          ns-prefix="ustmk"/>
  -->
</ipo>
```

The purpose of ipo.xml is to centralize IPO-specific namespace settings in one location apart from the Schematron source code, thereby keeping the Schematron source IPO-neutral.

1.7   Approved Exceptions Configuration

This section describes *optional* settings for approving exceptions to normal ST.96 rules checking. It is not required in order to use ST96XSDValidator.

Some rules violations may be decided to be acceptable. Rather than having acceptable violations forever appear on the summary report, it is possible to approve an exception to a ST.96 rule so that future runs of ST96XSDValidator will not report the violation.

For example, consider SD-43:

> Schemas MUST declare elements and attributes for date and time values using W3C schema date and time data types.

A heuristic used by ST96XSDValidator to identify elements and attributes related to dates and times is to look for the string "date" or the string "time" in the name of the component. After running ST96XSDValidator once, we found that this usually solid heuristic failed for the following XSDs:

```
Trademark/MarkMultimedia.xsd
Trademark/MarkMultimediaBag.xsd
Trademark/MarkMultimediaFileFormatCategory.xsd
Trademark/MarkMultimediaFileName.xsd
```

The occurrence of the "time" string in these components is a mere coincidence and not a sign that these components should be typed using the W3C schema time data type per SD-43.

To remove these XSDs from the violations list for SD-43, add an entry in the
ApprovedExceptions.xml file in the ST96XSDValidator/dev installation directory as follows:

```
<ApprovedExceptions>
    <ApprovedException ruleID="SD-43">
        <xsd subdir="Trademark" file="MarkMultimedia.xsd"/>
        <xsd subdir="Trademark" file="MarkMultimediaBag.xsd"/>
        <xsd subdir="Trademark"
file="MarkMultimediaFileFormatCategory.xsd"/>
        <xsd subdir="Trademark" file="MarkMultimediaFileName.xsd"/>
    </ApprovedException>
</ApprovedExceptions>
```

This entry says to ST96XSDValidator to suppress violation reports of SD-43 for the listed XSDs.

If xsd/@subdir is omitted, an XSD named xsd/@file found in any subdir will be considered to be
an approved exception.

Running ST96XSDValidator

1.8   From the command line

The preferred way to run ST96XSDValidator is from the command line because it
produces the highest quality output reports and can test validation via a unit test suite of over 350
automated assertions.

1.8.1   Checking Schema(s) in a Directory

To check an entire directory of XSDs (and the XSDs in the subdirectories within, recursively),
type the following command from the ST96XSDValidator/dev installation directory:

[Windows:]
```
 ST96XSDValidator "path_to_input_dir"
```
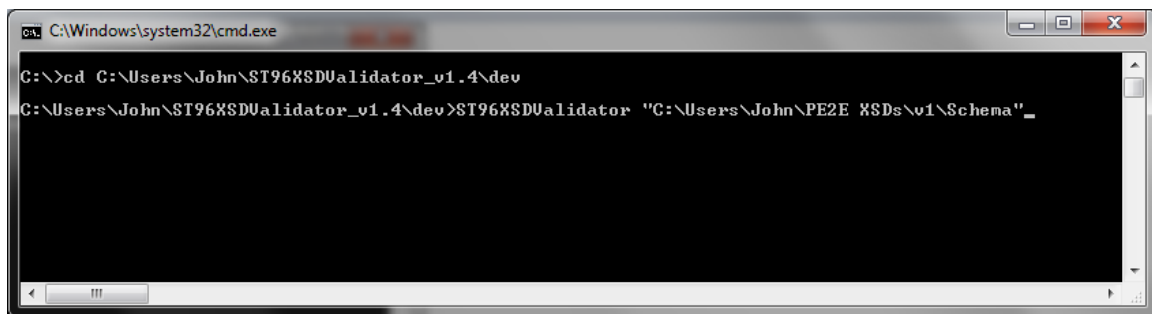
[Mac OS or Linux:]
```
  ./ST96XSDValidator.sh "path_to_input_dir"
```

For example, if the path to the input directory containing the XSDs to check is
C:\Users\John\ST96XMLSchema_V2_1, and executable installation directory is
c:\Users\John\ST96XSDValidator_v1.1, issue the following command:

[Windows:]
```
cd C:\Users\John\ST96XSDValidator_v1.4\dev
ST96XSDValidator "C:\Users\John\PE2E XSDs\v1\Schema"
```

[Mac OS or Linux:]
```
./ST96XSDValidator.sh "Users/John/PE2E XSDs/v1/Schema"
```
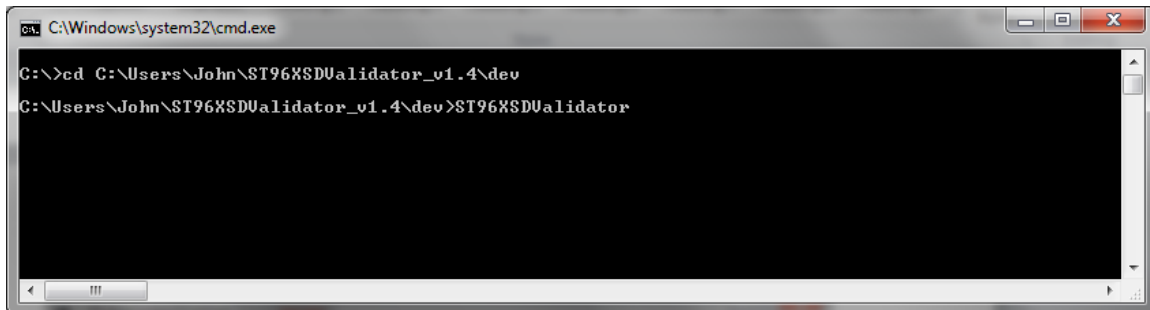
Alternatively, edit the build.xml file and point inputDir to the directory of XSDs to be checked:

```
<property name="inputDirCmdLine" value="path_to_input_dir"/>
```

Then issue the following command from the ST96XSDValidator/dev installation directory:

[Windows:]
```
ST96XSDValidator
```


[Mac OS or Linux:]
```
./ST96XSDValidator.sh
```

The results of the rules checking will be available in the output/summary.html file (as well as in a timestamped copy output/summary_DATE_TIME.html), an excerpt from which is show below:
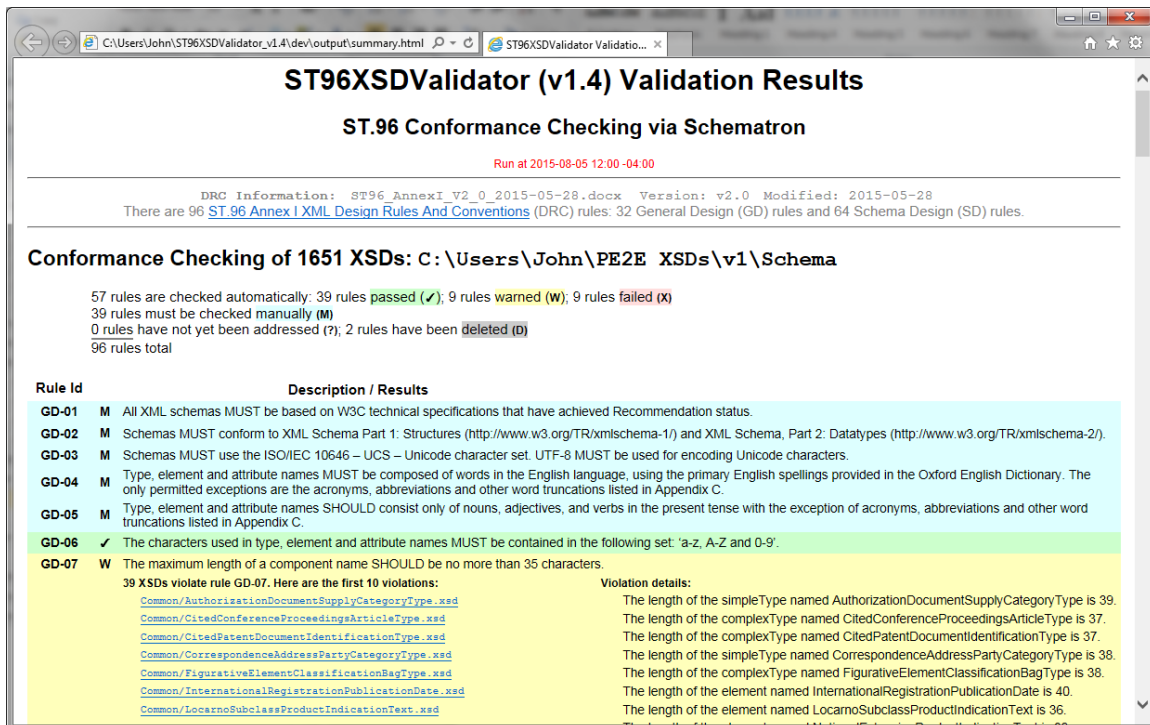
Figure 1: Validation Results Report

Technical Note: Developers familiar with Apache ant can bypass the xsdvalidate script and directly hit the ant target for generating a validation summary (summary). A list of other targets are available. Hit the "help" for details.

1.8.2   Unit Tests

Note: Unit tests are used by developers during development and testing of xsdvalidate. Most of the time, regular users will want to use the xsdvalidate command as described in Section 1.8.1.

To run the unit tests, type the following command from the ST96XSDValidator/dev directory:

[Windows:]
```
runtests
```



```
C:\Windows\system32\cmd.exe

C:\>cd C:\Users\John\ST96XSDValidator_v1.4\dev

C:\Users\John\ST96XSDValidator_v1.4\dev>runtests
```
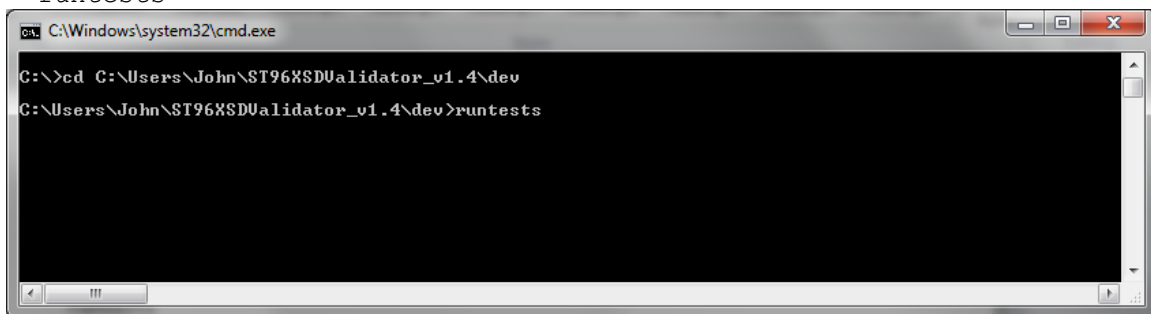
[Mac OS or Linux:]
```
./runtests
```

The results of the unit tests will be available in the output/status.html file, an excerpt from which is show below:
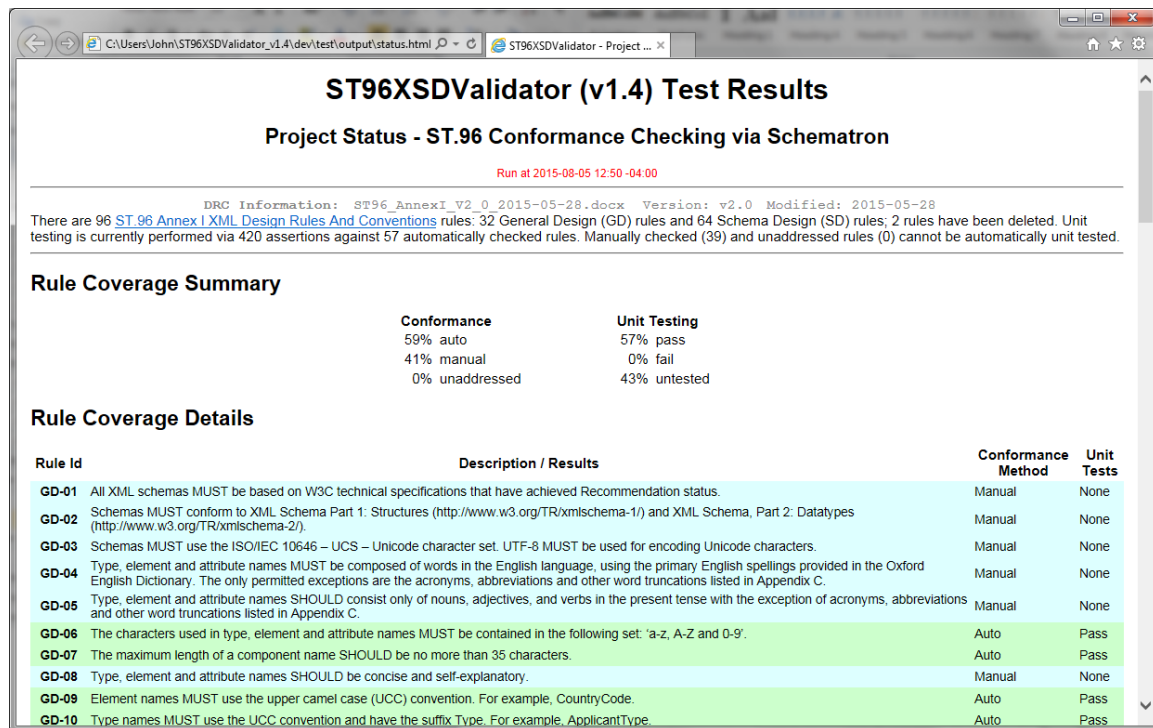

Figure 2: Project Status Report

Further details regarding unit testing of ST96XSDValidator can be found in Appendix B: Unit Testing Framework.

1.9   Using oXygen XML Editor

The oXygen XML Editor can validate a document using a Schematron schema[1]. This capability can be used in a limited manner to validate an XSD against the Schematron schema used in ST96XSDValidator.

> **Note**: The preferred way to run ST96XSDValidator is via the command line. Unlike the reports produced via ant build or Windows batch file, error and warning messages in oXygen show neither ST.96 rule identifier nor rule text. See Section **Error! Reference source not found.** for further advantages of running ST96XSDValidator via the command line.

---

[1] Note that only oXygen is discussed here because other major XML editors such as Altova XMLSpy do not currently have direct support for Schematron validation. However, note that a third-party, XML Buddy, has a plug-in that provides Schematron support in XMLSpy.

1.9.1   Checking a Single File

To check a single file using The oXygen XML Editor:

1. Open the XSD to be validated.
2. Choose the menu item **Document/Validate/Validate with...**
3. Choose the tab labeled **Schematron Schema**.
4. In the URL box, navigate to the location of the Schematron schema (ST96XSDValidator/dev/schemas/wipo_xsd.sch).
5. Click OK.

The resulting error messages will appear in a separate results box, as shown in **Error! Reference source not found.**. For many error messages, clicking on the error message will show the line in the document that triggered that message.
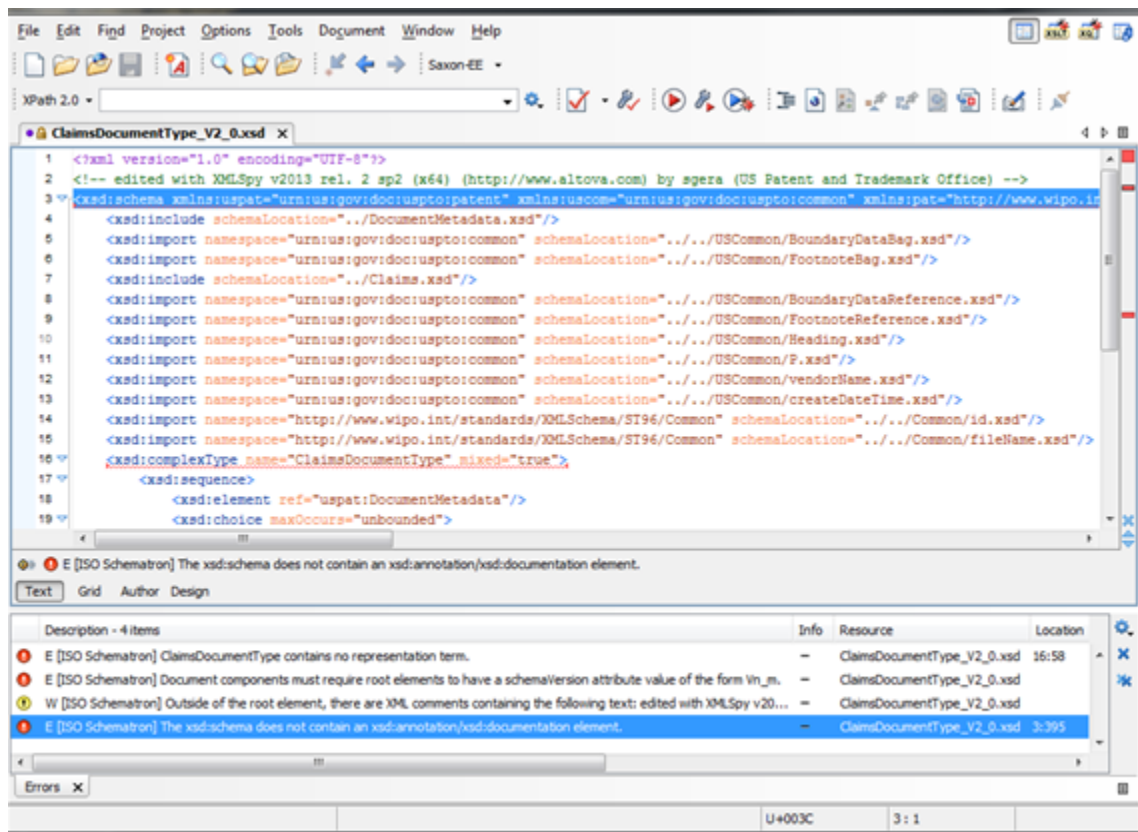

Figure 3: Errors/Warnings in The oXygen XML Editor

References

- WIPO Standard ST.96 Annex I - XML Design Rules and Conventions ([version 2.2](#))

- [Extensible Markup Language (XML) 1.0, Second Edition](#), Tim Bray et al., eds., W3C, 6 October 2000.

- [XML Schema Part 1: Structures](#), Henry S. Thompson et al., eds, W3C, 28 October 2004.

- [ISO Schematron reference implementation](#), (iso-schematron-xslt2.zip, for XSLT 2.0).

- [ISO Schematron tutorial: An introductory guide](#), Dave Pawson, Roger Costello, Florent Georges, 2007.

Appendix A: Rules Enforced by the Schematron

Table 1 lists the rules from the *XML Standards and Web Services using XML* document and indicates which ones are enforced automatically.

Table 1: Enforceability of XML rules

ST. 96 rule		Automation concern	Assumption to automate
GD-01	M	XSDs must use http://www.w3.org/2001/XMLSchema namespace anyway.	Not applicable.
GD-02	M	Validation against all of XSD 1.0 spec is very complicated.	Not applicable.
GD-03	M	Schematron and XSLT has not provision for querying encoding.	Not applicable.
GD-04	M	Referencing Oxford English Dictionary is difficult.	Not applicable.
GD-05	M	Knowing parts of speech and tense is difficult.	Not applicable.
GD-06	A	No concerns.	No assumptions required to automate.
GD-07	A	No concerns.	No assumptions required to automate.
GD-08	M	"Concise and self-explanatory" requires judgement.	Not applicable.
GD-09	A	Referencing dictionary is difficult. Knowing word boundaries is difficult.	Can detect these problems: • All capitals. • Wrong initial letter capitalization. Cannot detect these sorts of problems, for example: • BadelementName • BadElementname Names less than 3 characters are not checked.
GD-10	A	Referencing dictionary is difficult. Knowing word boundaries is difficult.	Can detect these problems: • All capitals. • Wrong initial letter capitalization. Cannot detect this sort of problems, for example: • BadnameType

GD-11	A	Referencing dictionary is difficult. Knowing word boundaries is difficult.	Can detect these problems: • All capitals. • Wrong initial letter capitalization. Cannot detect this sort of problem, for example: • bigbadAttribute
GD-12	A	Unintended matches could occur between a checked name and an expanded abbreviation in Appendix C.	Assume that occurrence of expanded Appendix C abbreviation anywhere in a name should be replace by corresponding abbreviation.
GD-13	A	Single-letter and double-letter abbreviations cannot be detected apart from regular single and double letters within words. Embedded multi-letter sequences can appear naturally within words without being abbreviations.	Don't automate single-letter or double-letter abbreviations. Accept some false-positive reports.
GD-14	M	Cannot detect all variations of an abbreviation.	Not applicable.
GD-15	M	"Meaningful" is subjective and requires human judgement.	Not applicable.
GD-16	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.
GD-17	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.
GD-18	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.
GD-19	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.

GD-20	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.
GD-21	M	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Not applicable.
GD-22	A	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Assume duplicated Representation Term is a sign of violation of this rule.
GD-23	A	Meaning of parts of names (Object Class, Property Term, Representation Term, Qualifier Term per ISO 11179 part 5) cannot be assessed automatically.	Assume presence of any word listed in Appendix B Representation Terms satisfies this rule.
GD-24	A	No concerns.	No assumptions required to automate.
GD-25	M	Plurality formation is too irregular to automate without dictionary access.	Not applicable.
GD-26	A	No concerns.	No assumptions required to automate.
GD-27	M	"Unless part of the business terminology" cannot be assessed automatically.	Not applicable.
GD-28	M	"Translated, changed or replaced" cannot be assessed automatically.	Not applicable.
GD-29	M	Detecting presence of 'article' or 'rule' could cause false positive errors.	Could accept any false positive errors. [*need decision here; try on PE2E?*]
GD-30	A	No concerns.	No assumptions required to automate.
GD-31 & GD-32	A	Cannot be sure whether a file is intended to be a draft.	Assume if "draft" or "D[0-9]" appears in file name, it is intended to be a draft; otherwise, assume it is not intended to be a draft.
GD-32	A	Cannot be sure whether draft is based on existing schema or is new.	Assume draft is based on existing schema.
SD-01	A	The xsd:import construct must be used to bring in files from other namespaces, so this rule appears to constrain nothing beyond that which is	If the common namespace prefix is declared on xsd:schema, then there must be an xsd:import of common.

		already constrained by XML Schema itself. Also, if a Patent, Trademark, or Design XSD does not require anything from Common, why must it import Common?	
SD-02	A	No concerns.	No assumptions required to automate.
SD-03	A	No concerns.	No assumptions required to automate.
SD-04	M	"Wherever applicable" cannot be assessed automatically.	Not applicable.
SD-05	M	"Maximum extent possible" cannot be assessed automatically.	Not applicable.
SD-06	A	No concerns.	No assumptions required to automate.
SD-07	A	No concerns.	No assumptions required to automate.
SD-08	A	No concerns.	No assumptions required to automate.
SD-09	M	Cannot detect changes to published namespaces.	Not applicable.
SD-10	A	No concerns.	No assumptions required to automate.
SD-11	A	Namespace qualifications[1] must be used to reference XML Schema constructs anyway, so this rule appears to constrain nothing beyond that which is already constrained by XML Schema itself. [1] Definition: A qualified name is a name subject to namespace interpretation [...] Syntactically, they are either prefixed names or unprefixed names. SD-15 already says that "Schemas SHOULD use "xsd" as a namespace prefix"	Check that namespace *prefixes* are used to reference XML Schema constructs. Where SD-15 says the "xsd" prefix SHOULD be used, here just say that some namespace prefix MUST be used.
SD-12	A	No concerns.	No assumptions required to automate.
SD-13	A	No concerns.	No assumptions required to automate.
SD-14	A	No concerns.	No assumptions required to automate.
SD-15	A	No concerns.	No assumptions required to automate.

SD-16	A	No concerns.	No assumptions required to automate.
SD-17	A	No concerns.	No assumptions required to automate.
SD-18	A	No concerns.	Common components are found in a "Common" subdirectory.

Applies to WIPO, not US-specific, XSDs. |
| SD-19 | A | No concerns. | Patent components are found in a "Patent" subdirectory.

Applies to WIPO, not US-specific, XSDs. |
| SD-20 | A | No concerns. | Trademark components are found in a "Trademark" subdirectory.

Applies to WIPO, not US-specific, XSDs. |
| SD-21 | A | No concerns. | Design components are found in a "Design" subdirectory.

Applies to WIPO, not US-specific, XSDs. |
SD-22	A	No concerns.	No assumptions required to automate.
SD-23	M	Cannot detect validation across versions of an XSD.	Not applicable.
SD-24	M	Cannot detect validation across versions of an XSD.	Not applicable.
~~SD-25~~	D	*Deleted*	
SD-26	A	How can flattened be determined?	Flatten schemas appear only in Document subdirectory.
SD-27	M	Cannot track version changes of XSDs.	Provide access to past versions of XSDs.
SD-28	M	Cannot track version changes of XSDs.	Provide access to past versions of XSDs.
SD-29	M	Cannot track version changes of XSDs.	Provide access to past versions of XSDs.
SD-30	A	No concerns.	No assumptions required to automate.
SD-31	A	No concerns.	No assumptions required to automate.

SD-32	M	Cardinality and granularity conformity across three past standards (ST.36, ST.66 or ST.86) is difficult to assess.	Not applicable.
SD-33	A	"Unless they are needed" is difficult to assess automatically.	Detect "Other" or "Undefined" without judging how needed they might be.
SD-34	A	Math formula are difficult to detect automatically. Type usage is difficult to ascertain because of the complexities of type and composition definition in XSD.	If an element name contains 'math' in any place in any case, conclude that it contains a math formula. Covering common XSD constructs and limiting recursive search to a depth of 6 is sufficient for a uses-type() function to answer correctly in most cases.
SD-35	A	Tables are difficult to detect automatically. Type usage is difficult to ascertain because of the complexities of type and composition definition in XSD.	If an element name contains 'table' in any place in any case, and the element name does not appear in the following list, conclude that the element contains a table: tablebody, tablegroup, tableimage, tabletitle, tabledata, tablecell, tabledatacell, tablefooter, tableheader, tableheadercell, or tablerow. Covering common XSD constructs and limiting recursive search to a depth of 6 is sufficient for a uses-type() function to answer correctly in most cases.
SD-36	M	There is no published list of approved industry-standard schemas.	Limit scope to ST.96 and assume that only OASIS Tables and MathML are approved to be imported.
SD-37	M	"Maximum extent possible" is difficult to assess.	Define an acceptable ratio of simple types to complex types. Could partially check by making sure that the major simple types (Language Code, two Country Codes) are used.
SD-38	M	What is it that is not allowed?	
SD-39	M	What is it that is not allowed?	
SD-40	A	Which elements are "used for representing IPOs and for priority and designated country/organization"?	Assume element name contains 'officecode'.
SD-41	A	Which elements are "used for the representation of the names of countries, dependencies, and other	Assume element name contains 'country'.

		areas of particular geopolitical interest"?	
SD-42	A	Which elements contain language codes?	Assume element name contains 'language'.
SD-43	A	Which elements contains date and time data?	Assume element name contains 'date' or 'time' respectively.
SD-44	A	Which elements contains currency data?	Assume element name contains 'currency' or 'money'.
SD-45	A	No concerns.	No assumptions required to automate.
SD-46	M	Judgement required.	Not applicable.
SD-47	M	What is it that is not allowed?	
SD-48	M	Difficult to automatically determine "non-business data".	Not applicable.
SD-49	A	No concerns.	No assumptions required to automate.
SD-50	A	No concerns.	No assumptions required to automate.
SD-51	A	No concerns.	No assumptions required to automate.
SD-52	A	No concerns.	No assumptions required to automate.
SD-53	M	What is it that is not allowed?	
SD-54	A	Cannot assess intent ("for accessibility").	Just check that xsd:any is not used.
SD-55	A	No concerns.	No assumptions required to automate.
SD-56	M	Cannot assess "appropriate" automatically.	
SD-57	M	Cannot assess "appropriate" automatically.	
SD-99	M	Implied from XSD already.	
SD-58	M	Cannot interpret documentation.	Could just detect presence of xsd:documentation on element and attribute definitions.
SD-59	A	No concerns.	No assumptions required to automate.
SD-60	M	Cannot interpret documentation.	Could try to parse documentation looking for patterns that look like enumerations. Has this been a problem? Are examples available?

SD-61	A	No concerns.	No assumptions required to automate.
SD-62	M	Cannot assess scope requirements.	
SD-63	A	No concerns.	All valid ST.3 codes are accepted.
SD-65	A	No concerns.	Geographical Indication components are found in a "GeographicalIndication" subdirectory. Applies to WIPO, not US-specific, XSDs.
SD-66	A	No concerns.	Copyright components are found in a "Copyright" subdirectory. Applies to WIPO, not US-specific, XSDs.

Appendix B: Unit Testing Framework Details

Each unit test is written specifically to a targeted, automatically checked, ST.96 rule. Each unit test is applied in isolation of all ST.96 rules other than the targeted rule. If the entire ST.96 Schematron file (ST96XSDValidator/dev/schemas/wipo_xsd.sch) were to be used *as-is* in unit testing, there could easily be cross-rule violations that would distract from the targeted test results or unnecessarily complicate creation of the targeted test. Instead, wipo_xsd.sch is automatically split into separate, self-contained Schema files, and the unit tests are applied to these separate Schema files in isolation.

The following subdirectories exist under ST96XSDValidator/dev/test in order to support unit testing of ST96XSDValidator on a per-ST.96-rule basis:
- **input/**
  Contains one subdirectory of XSD test cases for each ST.96 rule.
- **output/**
  Contains status.html, the combined and formatted results of all unit tests.
- **schemas/split_guidelines**
  Contains separate (automatically split) Schematron files for each ST.96 rule found in ST96XSDValidator/dev/schemas/wipo_xsd.sch.
- **schemas/split_guidelines_test**
  Contains a hand-written Schematron file for each automatically checked ST.96 rule.
  (Unit tests for ST96XSDValidator are themselves implemented in Schematron.)
- **svrl_conformance/**
  Contains results of applying the separated ST.96 Schematron files to each unit test XSD.
- **svrl_conformance_svrl/**
  Contains results of applying the hand-written unit test Schematron files (from schemas/split_guidelines_test) to the results (in svrl_conformance) of applying the separated ST.96 Schematron tests to the unit test input XSDs.
- **xsl/status.xsl**
  Hand-written XSLT file that formats results of unit testing (output/status.html).
- **xsl/split.xsl**
  Hand-written XSLT file that splits ST.96 Schematron into separate files per rule.
- **xsl/split_guidelines**

Contains XSLT compiled from schemas/split_guidelines.
- **xsl/split_guidelines_test**
  Contains XSLT compiled from schemas/split_guidelines_test.