ST.96 - ANNEX I

XML DESIGN RULES AND CONVENTIONS

Version 1.0

adopted by the Committee on WIPO Standards (CWS) at its second session on May 4, 2012

Table of Contents

1. INTRODUCTION

1.1 Overview

1. The *XML Design Rules and Conventions* provides an opportunity to promote the harmony across the three IP types and to facilitate data exchange amongst Industrial Property Offices (IPOs) by using reusable and interoperable XML schemas.

1.2 Scope

2. The scope of this document is to provide a comprehensive set of design rules and conventions for the creation and use of XML schemas and instances regarding all types of IP information to facilitate filing, processing, publication and data exchange among IP community.

1.3 How to use this document

3. This document is intended for use by IPOs, IP Data providers, and the wider IP community. The IP community should use this document as the point of reference for approved design rules and conventions with which all XML schemas must follow in order to be considered ST.96 compatible. IPOs can use this document as a guideline for developing their internal design rules. This document should be consulted prior to beginning development of a new XML schema or modifying an existing XML schema. After an XML schema has been developed, this document should be used to check the conformity of the schema with design rules.

1.4 Document structure

4. It is recommended to read this document in the order in which it was written. This guide is structured as follows:

- Section 1, Introduction, describes the general rules that apply throughout this document;
- Section 2, XML Design Conventions, defines high-level rules that apply to both schema and instance development efforts;
- Section 3, XML Schema Construct Conventions, defines specific design rules for using the W3C Schema specifications for creating XML schemas; and
- Section 4, Instance Design Rules, defines specific design rules for creating instances.

5. In addition, this guide contains four appendices:

- Appendix A, Summary of Design Rules, summarizes the design rules found in this document;
- Appendix B, Representation Terms, contains the definition of representation terms;
- Appendix C, List of Acronyms and Abbreviations, contains terms and abbreviations used in this document; and
- Appendix D, References, contains references to WIPO Standards, and other industry standards;

1.5 Terminology and notation

1.5.1 Key words

6. In general use,

- the term XML schema is a language for describing the structure and constraining the contents of XML documents; and
- the term W3C XML Schema refers to XML schemas that fully conform to the W3C XML Schema Definition Language suite of recommendations—*XML Schema Part 1: Structures and XML Schema; Part 2: Datatypes*.

- the term ST.96 compatible schema refers to a schema consistent with ST.96 Schema components and XML Design Rules and Conventions for Industrial Property (DRCs), i.e., Annex I of ST.96.
- the term ST.96 conformant schema refers to a compatible schema that has not been extended and that sustains constraints expressed by an ST.96 Schema;

7. In this document,

- the term Schema refers to XML schema defined in Annex III of WIPO Standard ST.96; and
- the term component refers to Type, element or attribute.

8. These DRCs contain certain keywords that have an explicit meaning. Those keywords, based on the definitions in *Request for Comments 2119* issued by the Internet Engineering Task Force, are as follows (non-capitalized forms of these words are used in the usual English sense);

- MUST: This word, or the terms REQUIRED or SHALL, means that the definition is an absolute requirement of the specification;
- MUST NOT: This phrase, or the phrase SHALL NOT, means that the definition is an absolute prohibition of the specification;
- SHOULD: This word, or the adjective RECOMMENDED, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course;
- SHOULD NOT: This phrase, or the phrase NOT RECOMMENDED, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label; and
- MAY: This word, or the adjective OPTIONAL, means that an item is truly optional. An implementation that does not include a particular option MUST be prepared to interoperate with another implementation that does include the option, though perhaps with reduced functionality. In the same vein, an implementation that does include a particular option MUST be prepared to interoperate with another implementation that does not include the option (except, of course, for the feature the option provides).

1.5.2 General notations

9. The following notations are used throughout this document:

- < >: Indicates a placeholder descriptive term that in implementation will be replaced with a specific instance value.
- " ": Indicates that the text included in quotes must be used verbatim in implementation.
- { }: Indicates that the items are optional in implementation.
- `Courier`: Indicates XML keywords, XML tag names and XML codes, appearing in `courier font`.

1.5.3 Rule identifiers

10. All design rules are normative. Design rules are identified through a prefix of [XX-nn].

(a) The value "XX" is a prefix to categorize the type of rule as follows:

– GD for general design rules;

– SD for schema design rules; and

– ID for instance design rules.

(b) The value "nn" indicates the sequential number of the rule.

For example, the rule identifier [GD-10] identifies the tenth general design rule.

2. XML DESIGN CONVENTIONS

2.1 General XML design rules

11. This section of the guide contains general, high-level XML design rules and guidelines that apply to all XML development efforts, rather than to a specific facet of XML technology. The general rules and guidelines, listed below, provide the common foundation for data and document development for IP information.

[GD-01] All XML schemas MUST be based on W3C technical specifications that have achieved Recommendation status.

[GD-02] Schemas MUST conform to
XML Schema Part 1: Structures (http://www.w3.org/TR/xmlschema-1/) and
XML Schema, Part 2: Datatypes (http://www.w3.org/TR/xmlschema-2/).

[GD-03] ISO/IEC 10646 – UCS – Unicode MUST be used for character set. UTF-8 MUST be used for encoding Unicode characters.

2.2 XML naming conventions

12. These conventions are necessary to ensure consistency, uniformity, and comprehensiveness in the naming and defining of all XML resources. These conventions are also suitable for file names.

2.2.1 Schema construct naming conventions

13. XML naming conventions of WIPO Standard ST.96 are based on the guidelines and principles described in document *ISO 11179 Part 5 - Naming and Identification Principles*. The name of Types, elements and attributes consists of the terms which are:

- Object Class refers to an activity or object within a business context and represents the logical data grouping or aggregation (in a logical data model) to which a Property belongs. The Object Class is expressed by an Object Class Term.
- Property Term identifies characteristics of the Object Class.
- Representation Term categorizes the format of the data element into broad types. Representation Terms listed in Appendix B to this document should be used for WIPO Standard ST.96.
- Qualifier Term is a word or words which help define and differentiate a data element from its other related data elements and may be attached to object class term or property term if necessary to make a name unique.

[GD-04] Type, element and attribute names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary, including office-specific names, except acronyms, abbreviations or other word truncations listed in Appendix C.

[GD-05] Type, element and attribute names MUST only contain nouns, adjectives and verbs in the present tense.

[GD-06] The characters used in Type, element and attribute names MUST be restricted to the following set: {a-z, A-Z and 0-9}.

[GD-07] The maximum length of a name SHOULD be 35 characters.

[GD-08] Type, element and attribute names SHOULD be concise and self-explanatory.

[GD-09] Element names MUST be in upper camel case (UCC) convention. For example, `CountryCode`.

[GD-10] Type names MUST be in UCC convention + suffix Type. For example, `ApplicantType`.

[GD-11] Attribute names MUST be in lower camel case (LCC) convention. For example, `currencyCode="EUR"`.

[GD-12] The acronyms and abbreviations listed in Appendix C MUST always be used instead of the complete extended name.

[GD-13] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear all in lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case.

[GD-14] Acronyms and abbreviations MUST appear in all upper case for all element and Type names.

[GD-15] Complex Type names SHOULD include a meaningful Object Class Term.

[GD-16] Association complex Type names SHOULD use a structure of Object Class of the associating complex Type, Property (nature of the association), and the Object Class of the associated complex Type and any Qualifiers. For example, `ApplicantResidenceAddress: Applicant` is Object Class of associating complex Type, `Residence` is Property, and `Address` is the Object Class of associated complex Type.

[GD-17] Simple Type and atomic element (it has no children) names SHOULD consist of the Object Class Term, Property Term, Representation Term and Qualifier Term.

[GD-18] An Object Class Term MUST always have the same semantic meaning throughout a namespace and MAY consist of more than one word. For example, `ContactInformation.`

[GD-19] A Property Term in a name MUST be unique within the context of an Object Class but MAY be reused across different Object Classes.

[GD-20] A Qualifier Term MAY be attached to an Object Class Term or a Property Term if necessary to make a name unique.

[GD-21] The Object Class Term MUST occupy the first (leftmost) position, the Property Term the next position and the Representation Term the last (rightmost) position in the name. Qualifier Term SHOULD precede the associated Object Class Term or Property Term.

[GD-22] If the Property Term ends with the same word as the Representation Term (or an equivalent word) then the Representation Term MUST be removed.

[GD-23] Representation Terms in Appendix B MUST be used for representation terms in component names.

[GD-24] Within a namespace, all Type, element and attribute names (whether declared locally or globally) MUST be unique.

[GD-25] Word(s) in a name SHOULD be in singular form unless the concept itself is plural. For example: `TotalMarkSeries`

[GD-26] The name of a collection SHOULD use the “Bag” suffix. For example, `EmailAddressBag` represents a collection of `EmailAddress`.

[GD-27] Connecting words like “and”, “of” and “the” SHOULD NOT be used in Type, element, and attribute names unless they are part of the business terminology. For example, `GoodsAndServices`.

[GD-28] Type, element and attribute names MUST NOT be translated, changed or replaced for any purpose.

[GD-29] Type and element names MUST NOT refer to article and rule numbers. For example, `PCTRule702C` for the PCT.

2.2.2 Schema file naming conventions

14. Schema file names and schema names are often paired. Schema file names rely on the corresponding schema names. For example, the file name of PostalAddressType.xsd is derived from the schema name `PostalAddressType`. Thus, schema file naming conventions are related to the rules for XML naming conventions in this document.

15. Schema file naming conventions are an important part of the schema versioning strategy. Schema file names must be implemented consistently to ensure that users can differentiate between schema versions. Thus, rules in this section are closely related to the rules in the *Schema Versioning* section in this document.

16. Schema file should have version information. It is recommended to include the version identifier in the schema file name. For Schema which is at the draft stage (draft Schema), revisions to the draft Schema are allowed. Draft Schemas must be denoted as such, in the Schema file name, putting the letter "D" and revision number.

[GD-30] The characters used in Schema file names MUST be restricted to the following set: {a-z, A-Z, 0-9, dash (-), and period (.)}.

[GD-31] Schema file name MUST consist of five parts with three delimiters, i.e.: <component name>"-""V"<major version number>"-"<minor version number>"."<file extension>. For example, EmailAddressType-V1-1.xsd, languageCode-V1-0.xsd, id-V1-2.xsd.

[GD-32] Draft Schema file name MUST consist of seven parts with four delimiters, i.e.: <component name>"-""V"<major version number>"-"<minor version number>"-""D"<revision number>"."<file extension>, for example, Contact-V1-0-D3.xsd.

2.3 Modularity strategy

17. WIPO Standard ST.96 relies extensively on modularity in schema design. Dividing the physical realization of schemas into multiple schema modules provides a mechanism whereby reusable components can be imported as needed without the need to import complete schemas. Therefore, ST.96 recommends avoiding the definition of all the elements and logical components in a single monolithic XML schema, which prevents the ability to share and reuse individual elements or logical components defined as a group in a schema.

2.3.1 Schema modules

18. Components defined in WIPO Standard ST.96 are categorized into Common Components, Patent Components, Trademark Components or Design Components. Common Components SHOULD be context-neutral (or business independent) and shared by, at least, two types of industrial property.

19. There are three different levels of component, i.e., Basic, Aggregate and Document Components. Common Components are classified into Basic Components and Aggregate Components, and Business Components into Basic, Aggregate and Document Components.

Level	Description
Basic Component	A Basic Component is used by one or more Aggregate Components. Basic Component is represented by an element. Basic Component refers to W3C Built-in Datatype or simple Type or complex Type with `xsd:simpleContent` definition. For example, `FirstName (string)`, `PhoneNumberCategory (PhoneNumberCategoryType)`.
Aggregate Component	An Aggregate Component is a collection of related Basic Components and/or other Aggregate Components that together convey a distinct business meaning, independent or not of any specific business context. It is represented by an element and a Type (complex Types incorporating multiple elements). Aggregate component refers to Basic Component definition, or other Aggregate Component definition or external standards specified in this document. For example, `Name (NameType)`, `Contact (ContactType)`.
Document Component	A Document Component is a top level component that defines the primary exchanged documents. It is represented by an element and a complex Type. Document Component refers to Basic Component definition, or Aggregate Component definition or external standard specified in this document. For example, `ApplicationBody (ApplicationBodyType)`, `MarkRecord (MarkRecordType)`.

20. The following example is for Basic Component declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:com="http://www.wipo.int/standards/XMLSchema/Common/0"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Common/0"
elementFormDefault="qualified" attributeFormDefault="qualified" version="0.7">…
      <xsd:include schemaLocation="sequenceNumber-V0-5.xsd"/>
      <xsd:include schemaLocation="TextType-V0-5.xsd"/>
      <xsd:complexType name="AddressLineType">
            <xsd:annotation>
                  <xsd:documentation>A line in an address. </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                  <xsd:extension base="com:TextType">
                        <xsd:attribute ref="com:sequenceNumber" use="required"/>
                  </xsd:extension>
            </xsd:simpleContent>
      </xsd:complexType>
</xsd:schema>
```

2.3.2 External schema reference

21. Each schema module resides in a distinct namespace. In this document, each schema module is defined in one of four namespaces which are, respectively, for Common Component schema modules, Patent Component schema modules, Trademark Component schema modules, and Design Component schema modules.

22. Two W3C Schema constructs, i.e., `include` and `import`, are used for external schema references. The `include` construct must be used when the including and included schemas have the same target namespace. The term 'including schema' refers to the schema that includes an external schema, while the term 'included schema' refers to the external schema. The `import` construct must be used when the including and included schemas have different target namespaces. This technique can be useful if an organization does not need to confine the use of constructs within schemas to only those that belong to a particular namespace.

23. Figure 1 describes how to use WIPO Standard ST.96 modularity model to develop schemas.

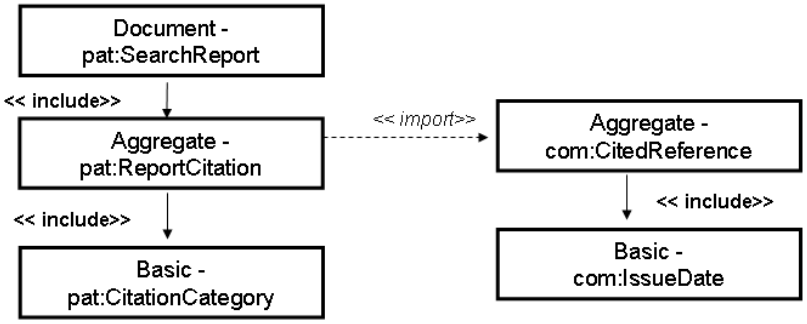

Figure 1. Example of Schema Modularity

[SD-01] Patent Component schema modules, Trademark Component schema modules, and Design Component schema modules MUST use the `import` construct to refer to Common Component schema modules.

[SD-02] Common Component schema module MUST NOT refer to Patent Component schema modules, Trademark Component schema modules and Design Component schema modules.

[SD-03] Patent Component, Trademark Component and Design Component schema modules MUST NOT refer to each other.

2.4 Reusability

24. Decoupling and cohesiveness are important design principles of XML schemas. Decoupling aims to minimize dependencies between elements, both in the instance document as well as in the schema. Cohesiveness aims to group together related pieces of data. Some design patterns have emerged that address decoupling and cohesiveness in XML schemas.

25. The design patterns also answer the question of how to vary the granularity of components (elements and Types) to be reused. The choice of an appropriate pattern is a critical step in the design phase of schemas. Thus, a schema design pattern should be decided before designing schemas.

26. The most common design patterns are Russian Doll, Salami Slice, Venetian Blind, and Garden of Eden. The patterns vary according to the number of their global components (elements or Types). To understand the design patterns, it is necessary to differentiate between a global component and a local component. A global component is an immediate child of the schema construct in the XML schema definition file. A local component is not an immediate child of the schema construct in the XML schema definition file. Global components are associated with the target namespace of the schema and may be reused in other schema. It is also important to understand that any element defined in the global namespace can be the root for a valid XML instance document adhering to the schema defined for that namespace.

27. Most real-world schemas adopt the Venetian Blind or the Garden of Eden as their pattern of choice because they are the most reusable. For the purposes of decoupling and cohesiveness, Venetian Blind may be better than Garden of Eden, but in terms of reusability, Garden of Eden may be better than Venetian Blind by using both Type and element. The most significant issue to a Type-based approach (Venetian Blind) is the risk of developing an inconsistent element vocabulary where elements are declared locally and allowed to be reused without regard to semantic clarity and consistency across Types. Therefore, WIPO Standard ST.96 recommends the Garden of Eden design pattern.

[SD-04] Existing Schemas MUST be used wherever applicable prior to creating new Schemas.

[SD-05] Schemas SHOULD use existing elements and Types to the maximum extent possible.

[SD-06] All Types, elements and attributes MUST be globally declared.

[SD-07] Schemas MUST NOT use `redefine` construct for redefinition.

2.5 Namespaces

28. Namespaces are used uniquely to identify elements and attributes with the same name when they are combined in a single document. Namespaces associate schema constructs with a conceptual space that defines a markup vocabulary. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace then the ambiguity between identically named elements or attributes can be resolved. Namespaces must be unique and persistent. In WIPO Standard ST.96, multiple-namespace configuration is recommended.

[SD-08] Schemas MUST use namespaces.

[SD-09] Published namespace declarations SHOULD NOT be changed.

2.5.1 Namespace declaration and qualification

[SD-10] Schemas MUST declare the W3C Schema namespace.

[SD-11] Schemas MUST use namespace qualifications for all W3C Schema constructs.

[SD-12] Schemas MUST use the URL-formatted namespace.

[SD-13] All schemas MUST have attributes `elementFormDefault` and `attributeFormDefault` with value “`qualified`”

29. For efficiency reasons, IPOs may choose to reduce or even eliminate namespaces in production XML processing systems. Nevertheless, for exchange, the above rules must be observed.

2.5.2 Namespaces in XML schema

30. Names of namespaces are important for schema versioning strategy. It is imperative that the namespace naming rules *should be followed* when designing schemas. Regardless of the minor version number of the given schema, the namespace must only contain the major version number. For example: http://www.wipo.int/standards/XMLSchema/Trademark/1

31. In the following example, a properly-formed namespace is shown:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/Common/1"
xmlns:xsd= "http://www.w3.org/2001/XMLSchema">
……..
</xsd:schema>
```

32. More information is available in the Schema Versioning section of this document.

[SD-14] The names for namespaces MUST have the following structure: http://www.wipo.int/standards/XMLSchema/<category>/<major>; where <category> is a token identifying the domain of Schema being used: `Common`, `Patent`, `Trademark`, or `Design`; and <major> is the major version number.

[SD-15] Schemas SHOULD use "`xsd`" as a namespace prefix for all W3C Schema constructs, "`com`" for all Common Component Schemas, "`pat`" for all Patent Component Schemas, "`tmk`" for all Trademark Component Schemas, "`dgn`" for all Design Component Schemas, "`tbl`" for OASIS Table Component Schemas and "`mathml`" for MathML Component Schemas.

2.5.3 Target namespaces

33. Declaring a target namespace in a schema ensures that all constructs within the schema will be associated with a namespace. Conversely, without a target namespace, constructs declared in the schema would not belong to any namespace. While a schema may have more than one declared namespace, only one namespace can be designated as the target namespace. In W3C recommendations, it is not required that a target namespace be declared in a schema. However, it is recommended to use a target namespace in WIPO Standard ST.96.

34. A Schema declares a target namespace which matches one of the declared namespaces. In accordance with the associated namespace, Schemas must declare one of the following four target namespaces which are defined in WIPO Standard ST.96:

- http://www.wipo.int/standards/XMLSchema/Common/<major>, for example,

```
<?xml version="1.0" UTF-8?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:com="http://www.wipo.int/standards/XMLSchema/Common/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Common/1"
elementFormDefault="qualified" attributeFormDefault="qualified">
  <xsd:include schemalocation="xxx.xsd"/>
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/Patent/<major>, for example,

```
<?xml version="1.0" UTF-8?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/Patent/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Patent/1"
elementFormDefault="qualified" attributeFormDefault="qualified">
  <xsd:include schemalocation="xxx.xsd"/>
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/Trademark/<major>, for example,

```
<?xml version="1.0" UTF-8?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tmk="http://www.wipo.int/standards/XMLSchema/Trademark/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Trademark/1"
elementFormDefault="qualified" attributeFormDefault="qualified">
<xsd:include schemalocation="xxx.xsd"/>
…
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/Design/<major>, for example,

```
<?xml version="1.0" UTF-8?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:dgn="http://www.wipo.int/standards/XMLSchema/Design/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Design/1"
elementFormDefault="qualified" attributeFormDefault="qualified">
<xsd:include schemalocation="xxx.xsd"/>
……
</xsd:schema>
```

[SD-16] Every Schema MUST declare its target namespaces using the `xsd:targetNamespace` attribute.

[SD-17] The Schema's `targetNamespace` MUST match the namespace name of one of the declared namespace qualifiers, but not the W3C qualifier.

[SD-18] Common Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Common/<major> as the target namespace.

[SD-19] Patent Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Patent/<major> as the target namespace.

[SD-20] Trademark Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Trademark/<major> as the target namespace.

[SD-21] Design Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Design/<major> as the target namespace.

2.5.4 Default namespaces

35. A default namespace reduces verbosity in a schema. However, declaration of a default namespace in a schema increases ambiguity because the omission of namespace prefixes makes it more difficult to identify the namespace where a construct belongs. Use of default namespaces can cause namespace coercion, making it impossible to discern the origin of schema constructs by examining the including schema. Therefore, use of default namespace is not recommended in WIPO Standard ST.96.

[SD-22] Schema SHOULD NOT use default namespace.

2.6 Schema versioning

36. Backward compatibility is an issue when existing schemas need to be changed. In general, backward compatibility exists if an XML instance document that validates against a previous version will continue to validate against the new version of the schema.

2.6.1 Major changes and minor changes

37. Changes to schemas are defined as major changes and minor changes. Major changes to schemas are those that are not backward compatible with previous versions of schemas whereas minor changes are backward compatible. New minor versions of schemas MUST be able to validate instance documents created with preceding minor versions of that schema with the same major version. However, instance documents should not be expected to validate against versions of a schema preceding the one they were created with. In addition, revisions are allowed for schema while the schema is at the draft stage.

2.6.1.1 Major versions

38. A major version of a Schema constitutes significant and/or non-backwards compatible changes. If any XML instance based on such an older major version Schema attempts validation against the newer version, it may experience validation errors. A new major version will be produced when significant and/or non-backward compatible changes occur, e.g.:

- removing or changing values in enumerations;
- changing of element names, Type names and attribute names;
- deleting or adding mandatory elements or attributes; and
- changing cardinality from optional to mandatory.

2.6.1.2 Minor versions

39. Within a major version of a WIPO Standard ST.96 Schema, there can be a series of minor or backward compatible, changes. The minor versioning of an ST.96 Schema determines its compatibility with an ST.96 Schema with preceding and subsequent minor versions within the same major version. The minor versioning scheme thus helps to establish backward and forward compatibility. Minor versions will only be increased when compatible changes occur, e.g.:

- adding values to enumerations,
- optional extensions, and
- adding optional elements and/or attributes.

[SD-23] New minor versions of Schemas MUST be able to validate all instance documents created with preceding minor versions of that Schema with the same major version.

[SD-24] The Schema major version MUST be incremented if the new draft of the Schema is not able to validate existing XML instance documents created with the current major version.

[SD-25] Schema SHOULD NOT have their minor versions changed unless the Schema or one of the imported/included Schemas is altered for minor changes.

[SD-26] For Schemas, schema file name, W3C `version` attribute, and namespace MUST all contain matching version information.

[SD-27] Schemas SHOULD have their versions changed when an included or imported schema is updated.

[SD-28] When any Schema construct is altered for major changes in a given namespace, all Schemas in the namespace MUST undergo a major version increment.

[SD-29] When creating a new Schema, the most recent versions of all the included or imported schemas SHOULD be used.

2.6.2 Schema versioning strategy

40. The following schemes work together to define the Schema versioning strategy:

a) Schema namespace naming conventions;

b) Schema file naming conventions;

c) use of the built-in XML schema `version` attribute; and

d) use of a user-defined `schemaVersion` attribute.

2.6.2.1 Namespace in schema versioning

41. According to the namespace naming rules set forth in this document, namespaces only contain a major version number. This is important to maintain compatibility among minor versions of schema. Since older minor version instance files must continue to validate against newer minor versions of the schema, the namespace must be retained among minor versions.

42. Namespace alone cannot be used to identify readily the exact version of a given schema or instance document. This is where other aspects of schema versioning become important, as described below.

2.6.2.2 File naming conventions in schema versioning

43. Version information can be included in the schema file name. According to the file naming rules set forth in this document, Schema filename contains both a major and a minor version number.

2.6.2.3 Built-in XML schema `version` attribute in schema versioning

44. The W3C Schema specification allows for a `version` attribute to be defined in the root element of a Schema. In WIPO Standard ST.96, Schemas must include both a major version number and a minor version number for each schema file using the W3C Schema `version` attribute.

For example,

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/Patent/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Patent/1"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
...
</xsd:schema>
```

[SD-30] The W3C Schema `version` attribute MUST include both a major version number and a minor version number for each schema file in the following format: <major version number>.<minor version number>.

2.6.2.4 User-defined `schemaVersion` attribute in schema versioning for XML instances.

45. The user-defined `schemaVersion` attribute is required in order to ensure that a given instance document directly refers to a specific schema version. This attribute will clearly define the schema version which the instance document targets. For example:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Patent/1"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/Patent/1"
elementFormDefault="qualified" attributeFormDefault="qualified" version="1.0">
<xsd:element name="SearchReport" type="SearchReportType"/>
<xsd:complexType name="SearchReportType"/>
   <xsd:attribute name="schemaVersion" type="xsd:String" use="required"/>
   ……
</xsd:complexType>
```

The corresponding XML instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<pat:SearchReport xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/Patent/1"
xsi:schemaLocation="http://www.wipo.int/standards/XMLSchema/Patent/1/Document
SearchReport-V1.0.xsd" schemaVersion="1.0">
……
</pat:SearchReport>
```

[SD-31] Schemas defined as Document components MUST define a required attribute named `schemaVersion` in their root element in the following format: <major version number>.<minor version number>.

2.7 Transformability with other WIPO XML Standards

46. Before adoption of WIPO Standard ST.96, WIPO Standards ST.36, ST.66 and ST.86 have already been used by IPOs. Therefore, it is one of primary concerns for WIPO Standard ST.96 to maintain transformability between ST.96 and ST.36, ST.66 or ST.86. In order to ensure that data can be processed satisfactorily for the business needs of IPOs and IP information suppliers, ST.96 seeks the necessary degree of transformability with WIPO Standards ST.36, ST.66, and ST.86.

47. Transformability means that the set of atomic information units (lowest level of granularity elements and attribute terminal leaves, typically an element without any child elements) that might be included in an XML instance which conforms to ST.96 is transformable into the atomic information units that might be included in an XML instance that conforms to ST.36, ST.66 or ST.86, and *vice versa*.

48. Enumeration lists should cater for the set of allowed values foreseen for the respective atomic information units available under WIPO Standards ST.36, ST.66 or ST.86. Enumeration lists should include the values, if needed, “Other”, meaning that the value provided in the source data is not present in the enumerated list, and, “Undefined”, meaning that the information was not present in the source data (XML instances conforming to ST.36, ST.66 and ST.86).

[SD-32] ST.96 Schemas SHOULD be developed with a similar level of cardinality and granularity to facilitate the transformability of ST.96 instances with instances of ST.36, ST.66 or ST.86.

[SD-33] Enumerations contained in Schemas SHOULD NOT include the values “Other” or “Undefined” unless they are needed.

2.8 Industry-standard schemas

49. Where appropriate to the content of a document, that is, where the content is not unique to the industrial property domain, use industry-standard schemas. Industry-standard schemas which are recommended in this document are OASIS XML table schema and MathML. The OASIS Table Schema is available at: http://www.oasis-open.org/docbook/xmlschema/1.0b1/calstbl.xsd which is a model schema and needed to be further specified for business need. In order to meet IP business, in ST.96 Schema, OASISTable.xsd is defined in basis of the model schema. For mathematical formulas, MathML, version 3, is recommended to use. Further information on MathML version3 is available at: http://www.w3.org/TR/MathML3.

50. In the case of chemical formulae or structures, there is neither W3C recommendation nor a widely agreed XML industry standard at this moment of the preparation of this document. There are, in fact, a large number of XML markup standards for a variety of chemical types, but none can be considered universally accepted. Thus this document does not recommend any XML standard for chemical data.

[SD-34] For mathematical formulas, MathML, version 3, SHOULD be used.

[SD-35] For tables, OASISTable.xsd which is based on the OASIS XML Schema SHOULD be used.

[SD-36] Industry-standard schemas SHOULD be incorporated by reference only and after prior approval by the Committee on WIPO Standards.

3. XML SCHEMA CONSTRUCT CONVENTIONS

3.1 Types definitions

51. Types represent the kind of information which elements and attributes can hold, for example, character strings or dates. Types and elements are often paired.

3.1.1 Simple types

52. The use of simple types increases data quality among XML applications because all applications that use simple types are subject to the same validations by XML processors. When the lexical format of a simple type is not suitable, schema developers can create their own datatypes using the W3C Schema Regular Expression syntax. Simple types include both W3C Built-in Datatypes and user-defined datatypes. In WIPO Standard ST.96, simple Types are defined as user-defined datatypes.

3.1.1.1 W3C built-in datatypes

53. W3C Built-in Datatypes are the data types that were defined by the W3C and included in the W3C Schema standard, for example, `date`, `Boolean`, `string` and `token`.

3.1.1.2 User-defined datatypes

54. One of the advantages of XML schemas is the ability to define user's datatypes. User-defined datatypes are based on the existing W3C Built-in Datatypes and can also be further derived from existing user-defined datatypes. User-defined datatypes can be derived in one of three ways: `restriction`, `list` and `union`.

[SD-37] Schemas SHOULD use simple Types to the maximum extent possible.

[SD-38] Simple Types SHOULD be used for defining the code lists using enumeration.

[SD-39] Simple Types MAY use `xsd:union` to enhance a code list.

[SD-40] WIPO Standard ST.3 two-letter codes MUST be used for representing IPOs and for priority and designated country/organization. For example, `PriorityCountryCode="EP"`.

[SD-41] ISO 3166-1-Alpha-2 Code Elements (2 letter country codes) MUST be used for the representation of the names of countries, dependencies, and other areas of particular geopolitical interest, on the basis of lists of country names obtained from the United Nations.

[SD-42] ISO 639-1 (2-Letter Language Codes) MUST be used for Language Codes.

[SD-43] Date and time values in Schema instances MUST conform to W3C XML Schema date and time data types.

[SD-44] ISO 4217-Alpha (3-Letter Currency Codes) MUST be used for Currency Codes.

[SD-45] The characters used in enumeration lists MUST be restricted to the following set: {a-z, A-Z, 0-9, period (.), comma (,), spaces, dash (-), ampersand (&) and underscore (_)}.

[SD-46] The values available in enumeration lists SHOULD be semantically sufficient, in English, and use as few characters as possible. The values SHOULD be drawn from common industrial property business language.

3.1.2 Complex types

55. Complex Types are user-defined Types that contain child elements and/or attributes.

[SD-47] Abstract Complex Types MAY be used.

3.2 Elements and attributes

56. Elements are the basic building blocks of an XML instance document and are represented by tags. Attributes are W3C Schema constructs associated with elements that provide further information regarding elements.

3.2.1 Element vs. attributes

57. One of the key schema design decisions is whether to represent a data element as an XML element or attribute. While elements can be thought of as containing data, attributes can be thought of as containing metadata. Once a data element has been made an attribute, it cannot be extended further. Therefore, attributes SHOULD only be used to describe information that cannot or will not be further extended or subdivided. Schemas SHOULD be designed so that elements are the main holders of industrial property information content in the XML instances. Attributes SHOULD hold ancillary metadata, i.e., simple items providing more information about the element.

[SD-48] Schemas SHOULD use attributes only when defining non-business data. For example, `sequenceNumber` attribute.

3.2.2 Elements

3.2.2.1 Cardinality of elements

58. The term *cardinality* is defined as the number of elements in a set.

59. Cardinality is indicated in a schema using the `minOccurs` and `maxOccurs` constraints in an element declaration; these constraints are also known as occurrence indicators. Occurrence indicators cannot appear within global element declarations.

[SD-49] Schemas SHOULD NOT use occurrence indicators for the default values (i.e., `minOccurs`="1"; `maxOccurs`="1").

3.2.2.2 Empty elements

60. An empty element is an element with no content, no child element, and no attribute. In general, the absence of an element in an XML schema does not have any particular meaning; it may indicate that the information is unknown, or not applicable, or the element may be absent for some other reason.

61. Although WIPO Standard ST.36 exploits empty elements as a kind of `Boolean` indicator that a condition was true or false, depending on the presence or absence of the element, that practice is deprecated in ST.96. Instead, elements should be created that explicitly assert either the true or the false condition, removing any doubt or ambiguity for both machine and human readers.

[SD-50] Empty elements MUST NOT be defined in Schemas except for line break.

3.2.3 Attributes

62. Attributes are W3C Schema constructs associated with elements that provide further information regarding elements. Unlike elements, attributes cannot be nested within each other, i.e., there are no sub-attributes. Therefore, attributes cannot be extended as elements can.

63. Cardinality for attributes differs from cardinality for elements. The `use` indicator can be specified for an attribute with one of the following values: "required", "optional" or "prohibited".

[SD-51] Schemas SHOULD NOT employ the `use` indicator for an attribute when the intended value is the default value (i.e., `use`= "`optional`").

3.2.4 Element and attribute grouping

64. Compositors are the W3C Schema constructs that group element declarations together. There are three kinds of compositors in the W3C Schema standard, i.e., `sequence`, `choice` and `all`.

65. The `sequence` compositor indicates that the elements declared inside it must appear in an XML instance document in the order declared. The `sequence` compositor allows element order enforcement.

66. The `choice` compositor indicates that only one of the elements declared within it can appear in an XML instance document.

67. In some cases, occurrence indicators in `sequence` and `choice` compositors allow flexibility of using `sequence`/ `choice` compositor, e.g., optional or multiple `sequence`/ `choice`, to avoid defining an extra level.

68. The `all` compositor indicates that the elements declared within it can appear only once in an XML instance document, in any order. With an `all` compositor, no element within it can appear more than once and all elements are optional if `minOccurs` attribute is set to 0. For data exchange, at least, one element should be mandatory. The use of the `all` compositor is, therefore, not recommended for Schemas.

[SD-52] Schemas SHOULD NOT use the `all` compositor.

[SD-53] Schemas MAY use occurrence indicators in `xsd:sequence` and `xsd:choice` compositors.

3.3 Extension and restriction

69. Some authorized techniques in W3C recommendations can affect data harmonization because they add flexibility to providing structured data. Those techniques are: extension, restriction and substitution groups.

3.3.1 Extension

70. Extension of complex Types does not affect data harmonization while extension of simple Types does because added components will not be exchanged; while deriving simple Types by `union` or `list` does impair interoperability because it allows new values that are not handled by the data exchange format.

[SD-54] A Schema MUST NOT contain a complex Type with `xsd:any` for extensibility.

3.3.2 Restriction

71. In the context of data harmonization, restriction of complex Types is acting on structure and can lead to compatibility problems blocking data exchange. In the case of simple Types, restriction is acting on values. No problem can be expected when exchanging data.

3.3.3 Substitution groups

72. Substitution groups allow a global element to replace another global element in an XML instance document without any further modifications to the schema. Substitution groups, however, do not promote the harmonization of element names. Harmonization is the key to interoperable data exchange and use of substitution groups is incompatible with harmonization.

[SD-55] Schemas MUST NOT use substitution groups.

3.4 Identity constraints

73. Like any storage system, an XML document needs to provide ways to identify and refer to pieces of the information it contains.There are two features that allow XML to do so with W3C XML Schema:

- `xsd:ID/xsd:IDREF/IDREFS` directly emulates the `ID`, `IDREF` and `IDREFS` attribute types from the XML DTDs; and
- `xsd:key/xsd:keyref/xsd:unique` was introduced to provide more flexibility through the use of XPATH expressions.

[SD-56] Schemas SHOULD use `xsd:key/xsd:keyref/xsd:unique` for identity constraints.

[SD-57] Schemas SHOULD NOT use `ID/IDREF/IDREFS` for identity constraints.

3.5 Schema documentation

74. The W3C schema specification allows for documentation to be included in schemas. This allows the schemas to contain embedded descriptions of each construct. Documentation is needed to communicate the context and usage of elements in schemas to the user who may not have the same background or business expertise that the schema developer has.

75. XML schemas may also contain comments (i.e., `<!-- comment -->`). It is recommended that schema developers avoid using this technique, especially if the purpose of the comment is to describe some aspect of a given schema construct.

76. Sometimes HTML-style comments are acceptable such as when the developer wishes to insert visual delimiters between constructs such as elements and Types.

[SD-58] All schemas SHOULD include schema construct documentation using the W3C documentation element. Documentation SHOULD only describe the element or Type and SHOULD NOT contain implementation details or other information not directly related to the meaning of the construct.

[SD-59] Comments (i.e., `<!-- comment -->`) SHOULD NOT be used in schema.

[SD-60] Documentation SHOULD NOT be substituted for code lists using enumeration.

3.5.1 Schema header documentation

77. Just as schema construct documentation adds clarity to a schema, schema header documentation enables information—the purpose, use, and contents of a schema—to be concentrated in a single place within a schema. As with schema construct documentation, the W3C Schema documentation element should be used for schema header documentation.

78. Table 1 lists the items that must be included in the header section of all Schemas.

Table 1. Header Documentation Information

Header item name	Description
“Last update”	Date of last update for the schema
“Contact Point”	Person to contact with questions about the schema
“Release Notes URL”	The URL which the release notes for the schema are published

The header item name should precede the actual contents of the header item, as in the following example:

```
<xsd:annotation>
  <xsd:documentation>
    Last update: 2011-05-28
    Contact point: xml.standards@wipo.int
    Release notes URL: http://www.wipo.int/standards/XMLSchema/ReleaseNotes.txt
  </xsd:documentation>
</xsd:annotation>
```

79. Schema header documentation allows a schema developer to easily discern the purpose, use, and contents of a schema. This information is also very helpful when a schema developer needs to select a schema to be used as a template in the creation of another schema.

[SD-61] Schemas MUST include schema header documentation containing references to “Last update”, “Contact point” and “Release notes URL”.

4. INSTANCE DESIGN RULES

80. Instance design rules deal with recommendations to follow when providing structured data.

4.1 Namespaces in XML instance documents

4.1.1 XML instance document validation

81. It is recommended to validate an XML instance document against a schema. Validation of an XML instance document ensures that its contents satisfy all requirements within the schema to which it validates. The validating system selects the schema based on its exact location as specified in the XML instance document.

82. Schema locations are required to be in the form of a URI. Schema locations are typically defined as URL based URIs because of resolvability limitations of URN based URIs

83. The schema location can be listed in the root element of the XML instance document, as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<com:Contact xmlns:com="http://www.wipo.int/standards/XMLSchema/Common/1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wipo.int/standards/XMLSchema/Common/1/Aggregat
e/Contact-V1-0.xsd">
…….
</com:Contact>
```

84. The `schemaLocation` attribute is a W3C Schema construct that associates an XML instance document with a schema. It is used only when a schema has a target namespace. In the example above, the "http://www.wipo.int/standards/XMLSchema/Common/1" namespace identifier is the target namespace of the Contact.xsd schema.

[ID-01] XML instances MUST be validated against the corresponding schema.

[ID-02] XML instance documents MUST use the `schemaLocation` attribute to describe the namespace and the schema name to which the XML instance conforms.

[ID-03] Each `xsi:schemaLocation` attribute declaration SHOULD contain a resolvable URL.

4.1.2 Namespace declaration and qualification in XML instance documents

85. As with Schemas, a namespace is declared in the root element of an XML instance document through the use of a namespace identifier along with a recommended namespace prefix. XML instance documents must use a namespace qualification for all elements which are defined in Annex III to WIPO Standard ST.96.

86. It should be noted that:

- the namespace identifier in an XML instance document must be the same as the namespace identifier for the target namespace in the schema; and
- the namespace prefix in an XML instance document does not need to be the same as the namespace prefix for the target namespace in the schema.

87. Elements and attributes in XML instance documents can be namespace qualified only if they belong to the target namespace of the schema that validates the XML instance document. Therefore, all global elements and attributes must be namespace qualified. However, the requirement for local elements and attributes that belong to the target namespace of the schema depends on the setting of a switch mechanism in the schema that uses the following two indicators:

- `elementFormDefault`, and
- `attributeFormDefault`.

88. The `elementFormDefault` indicator controls the namespace qualification of local elements, while the `attributeFormDefault` indicator controls the namespace qualification of local attributes. Both of these indicators appear as attributes of the root element of a schema, and each can have a value of `qualified` or `unqualified` (default).

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/Patent/1"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Patent/1"
elementFormDefault="qualified"
attributeFormDefault="qualified">
……
</xsd:schema>
```

89. These declarations require that all local elements and attributes in the target namespace of the schema be namespace qualified in an XML instance document unless the value of `form` attribute is altered.

[ID-04] Each declared namespace SHOULD be assigned to a recommended prefix, i.e., the prefix "`com`" for all Common Component Schema constructs, "`pat`" for all Patent Component Schema constructs, "`tmk`" for all Trademark Component Schema constructs, "`dgn`" for all Design Component Schema constructs, "`tbl`" for OASIS Table Component Schema constructs and "`mathml`" for MathML Component Schema constructs.

[ID-05] XML instance documents MUST NOT use default namespaces.

4.1.3 W3C schema instance namespace

90. The W3C Schema standard has its own namespace, referred to as the W3C Schema Instance namespace, which contains all W3C Schema constructs used in XML instance documents (`schemaLocation`, `noNamespaceSchemaLocation`, `type`, and `nil`). To use such constructs, the W3C Schema Instance namespace must be declared in the root element of an XML instance document using the namespace identifier "http://www.w3.org/2001/XMLSchema-instance":

```
<?xml version="1.0" encoding="UTF-8"?>
<com:Contact xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:com="http://www.wipo.int/standards/XMLSchema/Common/1">
……
</com:Contact>
```

91. Although user-defined, the prefix `xsi` is most often used in W3C Schema literature and references as the namespace prefix for W3C Schema Instance constructs.

[ID-06] XML instance documents SHOULD use `xsi` as a namespace prefix for all W3C Schema Instance constructs.

4.1.4 Namespace scope

92. Namespaces in XML instance documents have a scope of applicability in an XML instance document. The scope of a namespace applies to the declared element (which may be the root element) and all content within that element. A namespace can also be declared on an element other than the root element; this is known as a local namespace declaration.

93. Although processing efficiencies may be gained through the use of local namespace declarations, the ability to visually identify all namespaces declared in an instance document by examining the root element is more valuable.

[ID-07] XML instance documents MUST NOT use local namespace declarations.

4.2 External entities

94. Embedded images are a commonly used external entity, that is, a reference to an external image file is inserted in a document instance at the point where the image should be displayed when the instance is rendered. Embedded images are most often document parts that cannot be coded and stored using a character set. Embedded images MAY be drawings, chemical formulae, complex tables, undefined characters, etc.

95. An XML instance may refer to other XML file based on a DTD as external entity. In this case, the XML instance cannot be validated against corresponding XML schema. The possibility of reference to the XML file could be to use `xsd:anyURI` or `xsd:string` to point to the location of the external file of the XML instance. In XML instance, however, it is not recommended to refer to XML file based on DTD.

96. Images can be embedded in an XML instance as embedded binary images encoded in `base64Binary` that is the W3C Built-in Datatype as well as references to external image files, i.e., external entities. In this document, however, images must be referred to as external entities because embedded binary images can include harmful code, e.g., virus.

[ID-08] External entities that are images SHOULD conform to one of image formats recommended in WIPO Standard ST.96.

[ID-09] Images MUST be referred to as external files.

[ID-10] Sequence listings SHOULD follow WIPO Standard ST.25.

APPENDIX A - SUMMARY OF DESIGN RULES

General Design rules

Rule ID	Rule
[GD-01]	All XML schemas MUST be based on W3C technical specifications that have achieved Recommendation status.
[GD-02]	Schemas MUST conform to *XML Schema Part 1: Structures* (http://www.w3.org/TR/xmlschema-1/) and *XML Schema, Part 2: Datatypes* (http://www.w3.org/TR/xmlschema-2/).
[GD-03]	ISO/IEC 10646 – UCS – Unicode MUST be used for character set. UTF-8 MUST be used for encoding Unicode characters.
[GD-04]	Type, element and attribute names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary, including office-specific names, except acronyms, abbreviations or other word truncations listed in Appendix C.
[GD-05]	Type, element and attribute names MUST only contain nouns, adjectives and verbs in the present tense.
[GD-06]	The characters used in Type, element and attribute names MUST be restricted to the following set: {a-z, A-Z and 0-9}.
[GD-07]	The maximum length of a name SHOULD be 35 characters.
[GD-08]	Type, element and attribute names SHOULD be concise and self-explanatory.
[GD-09]	Element names MUST be in upper camel case (UCC) convention. For example, `CountryCode`.
[GD-10]	Type names MUST be in UCC convention + suffix Type. For example, `ApplicantType`.
[GD-11]	Attribute names MUST be in lower camel case (LCC) convention. For example, `currencyCode="EUR"`.
[GD-12]	The acronyms and abbreviations listed in Appendix C MUST always be used instead of the complete extended name.
[GD-13]	Acronyms and abbreviations at the beginning of an attribute declaration MUST appear all in lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case.
[GD-14]	Acronyms and abbreviations MUST appear in all upper case for all element and Type names.
[GD-15]	Complex Type names SHOULD include a meaningful Object Class Term.
[GD-16]	Association complex Type names SHOULD use a structure of Object Class of the associating complex Type, Property (nature of the association), and the Object Class of the associated complex Type and any Qualifiers. For example, `ApplicantResidenceAddress: Applicant` is Object Class of associating complex Type, `Residence` is Property, and `Address` is the Object Class of associated complex Type.
[GD-17]	Simple Type and atomic element (it has no children) names SHOULD consist of the Object Class Term, Property Term, Representation Term and Qualifier Term.
[GD-18]	An Object Class Term MUST always have the same semantic meaning throughout a namespace and MAY consist of more than one word. For example, `ContactInformation`
[GD-19]	A Property Term in a name MUST be unique within the context of an Object Class but MAY be reused across different Object Classes.
[GD-20]	A Qualifier Term MAY be attached to an Object Class Term or a Property Term if necessary to make a name unique.
[GD-21]	The Object Class Term MUST occupy the first (leftmost) position, the Property Term the next position and the Representation Term the last (rightmost) position in the name. Qualifier Term SHOULD precede the associated Object Class Term or Property Term.
[GD-22]	If the Property Term ends with the same word as the Representation Term (or an equivalent word) then the Representation Term MUST be removed.
[GD-23]	Representation Terms in Appendix B MUST be used for representation terms in component names.
[GD-24]	Within a namespace, all Type, element and attribute names (whether declared locally or globally) MUST be unique.
[GD-25]	Word(s) in a name SHOULD be in singular form unless the concept itself is plural. For example: `TotalMarkSeries`

Rule ID	Rule
[GD-26]	The name of a collection SHOULD use the “Bag” suffix. For example, EmailAddressBag represents a collection of `EmailAddress`.
[GD-27]	Connecting words like “and”, “of” and “the” SHOULD NOT be used in Type, element, and attribute names unless they are part of the business terminology. For example, `GoodsAndServices`.
[GD-28]	Type, element and attribute names MUST NOT be translated, changed or replaced for any purpose.
[GD-29]	Type and element names MUST NOT refer to article and rule numbers. For example, `PCTRule702C` for the PCT.
[GD-30]	The characters used in Schema file names MUST be restricted to the following set: {a-z, A-Z, 0-9, dash (-), and period (.)}.
[GD-31]	Schema file name MUST consist of five parts with three delimiters, i.e.: <component name>“-”“V”<major version number>“-”<minor version number>“.”<file extension>. For example, `SearchReport-V1-0.xsd`, `EmailAddressType-V1-1.xsd`, `languageCode-V1-0.xsd`, `id-V1-2.xsd`.
[GD-32]	Draft Schema file name MUST consist of seven parts with four delimiters, i.e.: <component name>“-”“V”<major version number>“-”<minor version number>“-”“D”<revision number>“.”<file extension>, for example, `Contact-V1-0-D3.xsd`.

Schema Design Rules

Rule ID	Rule
[SD-01]	Patent Component schema modules, Trademark Component schema modules, and Design Component schema modules MUST use the `import` construct to refer to Common Component schema modules.
[SD-02]	Common Component schema module MUST NOT refer to Patent Component schema modules, Trademark Component schema modules and Design Component schema modules.
[SD-03]	Patent Component, Trademark Component and Design Component schema modules MUST NOT refer to each other.
[SD-04]	Existing Schemas MUST be used wherever applicable prior to creating new Schemas.
[SD-05]	Schemas SHOULD use existing elements and Types to the maximum extent possible.
[SD-06]	All Types, elements and attributes MUST be globally declared.
[SD-07]	Schemas MUST NOT use `redefine` construct for redefinition.
[SD-08]	Schemas MUST use namespaces.
[SD-09]	Published namespace declarations SHOULD NOT be changed.
[SD-10]	Schemas MUST declare the W3C Schema namespace.
[SD-11]	Schemas MUST use namespace qualifications for all W3C Schema constructs.
[SD-12]	Schemas MUST use the URL-formatted namespace.
[SD-13]	All schemas MUST have attributes `elementFormDefault` and `attributeFormDefault` with value `"qualified"`
[SD-14]	The names for namespaces MUST have the following structure: http://www.wipo.int/standards/XMLSchema/<category>/<major>; where <category> is a token identifying the domain of Schema being used: `Common`, `Patent`, `Trademark`, or `Design`; and <major> is the major version number.
[SD-15]	Schemas SHOULD use "`xsd`" as a namespace prefix for all W3C Schema constructs, “`com`” for all Common Component Schemas, “`pat`” for all Patent Component Schemas, “`tmk`” for all Trademark Component Schemas, “`dgn`” for all Design Component Schemas, “`tbl`” for OASIS Table Component Schemas and “`mathml`” for MathML Component Schemas.
[SD-16]	Every Schema MUST declare its target namespaces using the `xsd:targetNamespace` attribute.
[SD-17]	The Schema’s `targetNamespace` MUST match the namespace name of one of the declared namespace qualifiers, but not the W3C qualifier.

Rule ID	Rule
[SD-18]	Common Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Common/<major> as the target namespace.
[SD-19]	Patent Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Patent/<major> as the target namespace.
[SD-20]	Trademark Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Trademark/<major> as the target namespace.
[SD-21]	Design Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/Design/<major> as the target namespace.
[SD-22]	Schema SHOULD NOT use default namespace.
[SD-23]	New minor versions of Schemas MUST be able to validate all instance documents created with preceding minor versions of that Schema with the same major version.
[SD-24]	The Schema major version MUST be incremented if the new draft of the Schema is not able to validate existing XML instance documents created with the current major version.
[SD-25]	Schema SHOULD NOT have their minor versions changed unless the Schema or one of the imported/included Schemas is altered for minor changes.
[SD-26]	For Schemas, schema file name, W3C `version` attribute, and namespace MUST all contain matching version information.
[SD-27]	Schemas SHOULD have their versions changed when an included or imported schema is updated.
[SD-28]	When any Schema construct is altered for major changes in a given namespace, all Schemas in the namespace MUST undergo a major version increment.
[SD-29]	When creating a new Schema, the most recent versions of all the included or imported schemas SHOULD be used.
[SD-30]	The W3C Schema `version` attribute MUST include both a major version number and a minor version number for each schema file in the following format: <major version number>.<minor version number>.
[SD-31]	Schemas defined as Document components MUST define a required attribute named `schemaVersion` in their root element in the following format: <major version number>.<minor version number>.
[SD-32]	ST.96 Schemas SHOULD be developed with a similar level of cardinality and granularity to facilitate the transformability of ST.96 instances with instances of ST.36, ST.66 or ST.86.
[SD-33]	Enumerations contained in Schemas SHOULD NOT include the values "Other" or "Undefined" unless they are needed.
[SD-34]	For mathematical formulas, MathML, version 3, SHOULD be used.
[SD-35]	For tables, OASISTable.xsd which is based on the OASIS XML Schema SHOULD be used.
[SD-36]	Industry-standard schemas SHOULD be incorporated by reference only and after prior approval by the Committee on WIPO Standards.
[SD-37]	Schemas SHOULD use simple Types to the maximum extent possible.
[SD-38]	Simple Types SHOULD be used for defining the code lists using enumeration.
[SD-39]	Simple Types MAY use `xsd:union` to enhance a code list.
[SD-40]	WIPO Standard ST.3 two-letter codes MUST be used for representing IPOs and for priority and designated country/organization. For example, `PriorityCountryCode="EP"`.
[SD-41]	ISO 3166-1-Alpha-2 Code Elements (2-letter country codes) MUST be used for the representation of the names of countries, dependencies, and other areas of particular geopolitical interest, on the basis of lists of country names obtained from the United Nations.
[SD-42]	ISO 639-1 (2-Letter Language Codes) MUST be used for Language Codes.
[SD-43]	Date and time values in Schema instances MUST conform to W3C XML Schema date and time data types.
[SD-44]	ISO 4217-Alpha (3-Letter Currency Codes) MUST be used for Currency Codes.
[SD-45]	The characters used in enumeration lists MUST be restricted to the following set: {a-z, A-Z, 0-9, period (.), comma (,), spaces, dash (-), ampersand (&) and underscore (_)}.

Rule ID	Rule
[SD-46]	The values available in enumeration lists SHOULD be semantically sufficient, in English, and use as few characters as possible. The values SHOULD be drawn from common industrial property business language.
[SD-47]	Abstract Complex Types MAY be used
[SD-48]	Schemas SHOULD use attributes only when defining non-business data. For example, `sequenceNumber` attribute
[SD-49]	Schemas SHOULD NOT use occurrence indicators for the default values (i.e., `minOccurs="1"`; `maxOccurs="1"`).
[SD-50]	Empty elements MUST NOT be defined in Schemas except for line break.
[SD-51]	Schemas SHOULD NOT employ the `use` indicator for an attribute when the intended value is the default value (i.e., `use`= "optional").
[SD-52]	Schemas SHOULD NOT use the `all` compositor.
[SD-53]	Schemas MAY use occurrence indicators in `xsd:sequence` and `xsd:choice` compositors.
[SD-54]	A Schema MUST NOT contain a complex Type with `xsd:any` for extensibility.
[SD-55]	Schemas MUST NOT use substitution groups.
[SD-56]	Schemas SHOULD use `xsd:key/xsd:keyref/xsd:unique` for identity constraints.
[SD-57]	Schemas SHOULD NOT use `ID/IDREF/IDREFS` for identity constraints.
[SD-58]	All schemas SHOULD include schema construct documentation using the W3C documentation element. Documentation SHOULD only describe the element or Type and SHOULD NOT contain implementation details or other information not directly related to the meaning of the construct.
[SD-59]	Comments (i.e., `<!--comment -->`) SHOULD NOT be used in schema.
[SD-60]	Documentation SHOULD NOT be substituted for code lists using enumeration.
[SD-61]	Schemas MUST include schema header documentation containing references to last update, contact point and release notes URL.

Instance Design rules

Rule ID	Rule
[ID-01]	XML instances MUST be validated against the corresponding schema.
[ID-02]	XML instance documents MUST use the `schemaLocation` attribute to describe the namespace and the schema name to which the XML instance conforms.
[ID-03]	Each `xsi:schemaLocation` attribute declaration SHOULD contain a resolvable URL.
[ID-04]	Each declared namespace SHOULD be assigned to a recommended prefix, i.e., the prefix "`com`" for all Common Component Schema constructs, "`pat`" for all Patent Component Schema constructs, "`tmk`" for all Trademark Component Schema constructs, "`dgn`" for all Design Component Schema constructs, "`tbl`" for OASIS Table Component Schema constructs and "`mathml`" for MathML Component Schema constructs.
[ID-05]	XML instance documents MUST NOT use default namespaces.
[ID-06]	XML instance documents SHOULD use `xsi` as a namespace prefix for all W3C Schema Instance constructs.
[ID-07]	XML instance documents MUST NOT use local namespace declarations.
[ID-08]	External entities that are images SHOULD conform to one of image formats recommended in WIPO Standard ST.96.
[ID-09]	Images MUST be referred to as external files.
[ID-10]	Sequence listings SHOULD follow WIPO Standard ST.25.

APPENDIX B - REPRESENTATION TERMS

Term	Definition	Data Type
Amount	A monetary value.	`AmountType`
Category	A specifically defined division or subset in a system of classification in which all items share the same concept of taxonomy.	`xsd:token`
Code	A combination of one or more numbers, letters, or special characters, which is substituted for a specific meaning. Represents finite, predetermined values.	`xsd:token`
Date	The notion of a specific point in time, expressed by year, month, and day.	`DateType`
Identifier	A combination of one or more integers, letters, special characters which uniquely identifies a specific instance of a business object, but which may not have a readily definable meaning.	`xsd:token`
Indicator	A signal of the presence, absence, or requirement of something. Recommended values are `Y`, `N`, and, "?" if needed.	`xsd:boolean`
Measure	A measure is a numeric value determined by measuring an object along with the specified unit of measure. `MeasureType` is used to represent a kind of physical dimension such as temperature, length, speed, width, weight, volume, latitude of an object. More precisely, `MeasureType` should be used to measure intrinsic or physical properties of an object seen as a whole.	`MeasureType`
Name	The designation of an object expressed in a word or phrase.	`xsd:string`
Number	A numeral or string of numerals expressing label, value, quantity or identification.	`xsd:positiveInteger` or `xsd:string`
Percent	A number which represents a part of a whole, which will be divided by 100.	`xsd:decimal`
Quantity	A quantity is a counted number of non-monetary units, possibly including fractions. `QuantityType` is used to represent a counted number of things. `QuantityType` should be used for simple properties of an object seen as a composite or collection or container to quantify or count its components. `QuantityType` should always express a counted number of things, and the property will be such as total, shipped, loaded, stored.	`QuantityType`
Rate	A quantity or amount measured in relation to another quantity or amount.	`xsd:decimal`
Text	An unformatted character string, generally in the form of words. (includes: Abbreviation, Comments.)	`xsd:string`
Time	A designation of a specified chronological point within a period.	`xsd:time`
Timestamp	The captured date and time of an event when it occurs.	`xsd:datetime`
URI	The Uniform Resource Identifier that identifies where the file is located.	`xsd:anyURI`

APPENDIX C - LIST OF ACRONYMS AND ABBREVIATIONS

ALT	Alternate Text for image
B	Bold
BioDeposit	Biological Deposit
Br	Break
DD	Definition Description
DL	Definition List
DOI	Digital Object Identifier
DT	Definition Term
DTD	Document Type Definition
I	Italic
ID	Identifier for system identification
IDREF	Identifier Reference
IP	Industrial Property
IPC	International Patent Classification
IPO	Industrial Property Office
IPR	Industrial Property Right
ISO	International Organization for Standardization
LCC	Lower Camel Case
LI	List Item
NPL	Non Patent Literature
O	Over score
OCR	Optical character recognition
OL	Ordered List
P	Paragraph
Pre	Preformatted text
Sub	Subscript
Sup	Superscript
SWIFT	Society for Worldwide Interbank Financial Telecommunication
U	Underlined
UCC	Upper Camel Case
UL	Unordered List
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WIPO	World Intellectual Property Organization
XML	eXtensible Markup Language

APPENDIX D - REFERENCES

WIPO Standards

- WIPO Standard ST.3: Two-letter codes for the representation of States, other entities and organizations
- WIPO Standard ST.25: Presentation of nucleotide and amino acid sequence listings
- WIPO Standard ST.36: Processing of patent information using XML
- WIPO Standard ST.66: Processing of trademark information using XML
- WIPO Standard ST.67: Electronic management of the figurative elements of trademarks
- WIPO Standard ST.86: Processing of industrial design information using XML

Industry Standards

- W3C XML Schema Definition Language (XSD) suite of recommendations—*XML Schema Part 1: Structures and XML Schema; Part 2: Types*
- Request for Comments 2119 issued by the Internet Engineering Task Force
- ISO/IEC 10646- Information technology — Universal multiple-octet coded character set (UCS)
- ISO 11179, Information Technology -- Metadata registries (MDR), Part 5: Naming and identification principles
- ISO 3166-1:2006, Codes for the representation of names of countries and their subdivisions – Part 1: Country codes
- ISO 639-1:2002, Codes for the representation of names of languages – Part 1: Alpha-2 code
- ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times
- ISO 4217:2008, Codes for the representation of currencies and funds
- OASIS XML Table Schema: http://www.oasis-open.org/docbook/xmlschema/1.0b1/calstbl.xsd
- MathLab: MathML, version 3. See http://www.w3.org/TR/MathML3 for a complete description

[End of Annex I]