ST.96 - ANNEX I

XML DESIGN RULES AND CONVENTIONS

Version 2.1

Revision approved by the XML4IP Task Force of the Committee on WIPO Standards (CWS) on December 11, 2015

Table of Contents

1. INTRODUCTION

1.1 Overview

1. The *XML Design Rules and Conventions* provide an opportunity to promote harmony across all types of industrial property (IP) as defined in WIPO Standard ST.96 and to facilitate data exchange amongst Industrial Property Offices (IPOs) by providing 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 covering all types of IP information to facilitate filing, processing, publication and data exchange among the 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 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

<u>1.5.1 Key words</u>

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*.

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 a type, element or attribute.

8. The keywords MUST, MUST NOT, SHALL, SHOULD, SHOULD NOT, and MAY, when they appear in this document, are to be interpreted as described in the “Definitions and Terminology” Section of WIPO Standard ST.96. Non-capitalized forms of these words are used in the regular English sense.

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 font`: Indicates XML keywords, XML tag names and XML codes.

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 next available number in the sequence of a specific rule type. It should be noted that the number does not mean the position of the Rule, in particular, for a new rule. A new Rule will be placed in the relevant context. For example, the rule identifier [GD-40] identifies the fortieth general design rule. The Rule [GD-40] can be placed between Rules [GD-20] and [GD-21] instead of following [GD-39] if that is the most appropriate location for this rule.

(c) The rule identifier of the deleted Rule will be kept while the Rule will be replaced with the text “Deleted”.

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] Schemas MUST use the ISO/IEC 10646 – UCS – Unicode 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.

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 consist of the following terms:

- 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 other related data elements and may be attached to an 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. The only permitted exceptions are the acronyms, abbreviations and other word truncations listed in Appendix C.

[GD-05] Type, element and attribute names SHOULD consist only of nouns, adjectives, and verbs in the present tense with the exception of acronyms, abbreviations and other word truncations listed in Appendix C.

[GD-06] The characters used in type, element and attribute names MUST be contained in the following set: ‘a-z, A-Z and 0-9’.

[GD-07] The maximum length of a component name SHOULD be no more than 35 characters.

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

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

[GD-10] Type names MUST use the UCC convention and have the suffix Type. For example, `ApplicantType`.

[GD-11] Attribute names MUST use the 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 in all lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case as listed in Appendix C.

[GD-14] Acronyms and abbreviations MUST appear as listed in Appendix C for element and type names.

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

[GD-16] When we need two Object Class terms in a given component name due to the nature of business entity, association complex type can be used. Names of an association complex type SHOULD use a structure of Object Class term of the associating complex type, a Property term that describes the nature of the association, and the Object Class of the associated complex type. Qualifiers may precede the Object Class and Property Term. For example, in `ApplicantResidenceAddress, Applicant` is Object Class of associating complex type, `Residence` is the Property, and `Address` is the Object Class of associated complex type.

[GD-17] Names of basic elements (those based on a simple or complex type that do not permit children) SHOULD consist of an Object Class Term, Property Term, and where applicable a Representation Term. Qualifier Terms may precede the Object Class term and Property 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] When a name contains an Object Class Term, a Property Term, and a Representation Term, the Object Class Term MUST precede the Property Term and the Property Term MUST precede the Representation Term. A 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] Where a representation term is required, the Representation Terms in Appendix B MUST be used for representation terms in Basic component names.

[GD-24] Within a namespace, all type, element and attribute names 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 an element or a type which contains a collection of contextually related components SHOULD have the "Bag" suffix. For example, `EmailAddressBag` represents a collection of `EmailAddress` elements.

[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.

[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 filenames and schema names are often paired. Schema filenames rely on the corresponding schema names. For example, the filename 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. A schema file MAY have version information. A schema which is at the draft stage may be revised. Draft Schemas must be denoted as such, in the Schema filename, putting the letter "D" and revision number.

[GD-30] The characters used in Schema filenames MUST belong to the following set: 'a-z, A-Z, 0-9, underscore "_", and period ".".

[GD-31] A Schema filename MUST consist of two compulsory parts with one delimiter and optional version information with two additional delimiters, i.e.: <component name>{"_""V"<major version number>"_"<minor version number>"}."<file extension>;. For example, EmailAddressType.xsd, languageCode.xsd, ApplicationBody_V1_0.xsd.

[GD-32] A draft Schema filename MUST consist of four compulsory parts with two delimiters and optional version information with two additional delimiters, i.e.: <component name>{"_""V"<major version number>"_"<minor version number>"}_""D"<revision number>"."<file extension>, for example, Contact_D3.xsd, TrademarkApplication_V1_1_D1. If a draft schema is based on an existing schema and has version information in its filename, the major and minor version numbers in the draft schema filename SHOULD be the same as specified in the schema file that the draft schema is based on. If a draft schema is new, the major version number in the draft schema filename SHOULD be the same number as specified in the corresponding namespace and a minor version number in the draft schema file SHOULD be zero "0".

2.3 Modularity strategy

16. 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 instead of complete schemas. ST.96 therefore recommends avoiding the definition of all the elements and logical components in a single monolithic XML schema, which reduces the ability to share and reuse individual elements or logical components defined as a group in a schema.

2.3.1 Schema modules

17. 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) or shared by, at least, two types of industrial property. Exceptionally, taking into account the evolution of ST.96 in the future, some components such as “P” element could be defined as a Common component.

18. There are three different kinds of component, i.e., Basic, Aggregate and Document; Document components, made up of Basic, Aggregate and/or other Document components, are defined to meet a business need. Basic and Aggregate components are defined in the Common Domain. Basic, Aggregate, and Document components are defined in Patent, Trademark and Design Domains.

Level	Description
Basic Component	A Basic Component refers to a W3C Built-in Datatype, simple Type or complex Type with `xsd:simpleContent` definition. It is represented by an element or an attribute. 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 of any specific business context. It is represented by an element or a complex type. For example, `Name` (`NameType`), `Contact` (`ContactType`).
Document Component	A Document Component is a grouping of components comprising Basic, Aggregate, and/or other Document components, which supports a specific aspect of an IP business such as filing, examination, publication, issuance, and data exchange. For example, `ApplicationBody,` based on `ApplicationBodyType`~~,~~ represents a patent specification and can be used for the filing and publication of a patent specification.

2.3.2 External schema reference

19. Each schema module resides in a distinct namespace. In ST.96, 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.

20. 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.

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

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

[SD-03] A Patent Component, Trademark Component or Design Component schema module MAY make reference to Common Component schema modules, schemas in the same Component module, and approved industry-standard schemas, but MUST NOT refer to other Component schema modules. For example, Patent schemas MUST NOT refer to Trademark schemas and vice versa.

2.4 Reusability

21. 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.

22. 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.

23. 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.

24. 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 global types and elements. 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. For this reason 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 elements and types defined in existing schemas to the maximum extent possible.

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

[SD-07] Schemas MUST NOT use `xsd:redefine` construct.

2.5 Namespaces

25. Namespaces are used to uniquely 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 URI-formatted namespace.

[SD-13] All schemas MUST have attributes `elementFormDefault` and `attributeFormDefault` with value “`qualified`” in the root `xsd:schema` element.

26. 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

27. WIPO Standard ST.96 defines four namespaces: Common, Patent, Trademark and Design. These namespaces are symbolic and constant across releases.

28. The following example shows, a properly-formed namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd= "http://www.w3.org/2001/XMLSchema">
………
</xsd:schema>
```

[SD-14] The names for namespaces MUST have the following structure: `http://www.wipo.int/standards/XMLSchema/ST96/<category>`; where <category> is a token identifying the domain of Schema being used: `Common`, `Patent`, `Trademark`, or `Design`.

[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

29. 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.

30. 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/ST96/Common, 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"
targetNamespace="http://www.wipo.int/standards/XMLSchema/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version=”V2_0”>
…
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/ST96/Patent, 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/ST96/Patent"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified" attributeFormDefault="qualified" version=”V2_0>
<xsd:include schemaLocation="xxx.xsd"/>
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/ST96/Trademark, 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/ST96/Trademark"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Trademark"
elementFormDefault="qualified" attributeFormDefault="qualified" version=”V2_0>
…
</xsd:schema>
```

- http://www.wipo.int/standards/XMLSchema/ST96/Design, 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/ST96/Design"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version=”V2_0>
…
</xsd:schema>
```

[SD-16] Every schema MUST declare the target namespace using the `xsd:targetNamespace` attribute.

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

[SD-18] Common Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Common as the target namespace.

[SD-19] Patent Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Patent as the target namespace.

[SD-20] Trademark Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Trademark as the target namespace.

[SD-21] Design Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Design as the target namespace.

2.5.4 Default namespaces

31. 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 a default namespace is not recommended in WIPO Standard ST.96.

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

2.6 Schema versioning

32. 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

33. 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

34. 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

35. 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 incremented when compatible changes occur. For example,

- 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 previous minor versions of that Schema with the same major version.

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

[SD-25] *Deleted*

[SD-26] *Deleted*

[SD-27] The major and/or minor version number of a schema SHOULD be changed when an included or imported schema is updated.

[SD-28] When any schema construct is altered, all schemas in the release MUST undergo the same version number 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

36. The following conventions work together to define the schema versioning strategy:

a) schema release folder structure and schema file naming conventions;

b) use of the built-in XML schema `version` attribute for all components;

c) use of a user-defined `schemaVersion` attribute at document instance level; and

d) use of a user-defined `ipoVersion` attribute at document instance level.

2.6.2.1 Folder and file naming conventions in schema versioning

37. Version information SHOULD be included in the folder, document level schema filename and flattened schema filename. According to the file naming rules defined in this document, schema release folder, document level schema filename and flattened filename contain both a major and a minor version number. In the ST.96 schema release folder structure, the release version number is followed by component types, i.e., Common, Patent, Trademark, Design and External Standards. For example, the version 2.0 of `Contact` component is placed in the folder, `ST96/V2_0/Common`.

[SD-64] Schema release folder, a document level schema filename and a flattened schema filename MUST contain matching version information comprising the major and minor version number.

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

38. 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/ST96/Patent"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="
V2_0">
...
</xsd:schema>
```

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

2.6.2.3 User-defined `st96Version` and `ipoVersion` attributes in schema versioning for XML instances.

39. The user-defined `st96Version` attribute is required in order to ensure that a given instance document directly refers to a specific version of ST.96 XML Schema. This attribute will clearly define the ST.96 schema version which the instance document targets or an implemented XML schema refers to. This attribute should be declared as required with a fixed value.

40. The user-defined `ipoVersion` attribute is required in order to ensure that a given instance document directly refers to a specific version of an IPO’s implemented schema. This attribute will clearly define the IPO’s schema version which the instance document targets. This attribute should be declared as optional and not populated when ST.96 schema components are used unchanged.

41. Considering backward and forward compatibility and data exchange among IPOs for which different minor versions are used, the major and minor version number should be provided. The following examples show how the attributes can be defined in ST.96 XML Schema, office’s implementation schema and XML instance.

42. The following example shows how the attributes of `st96Version` and `ipoVersion` should be defined in ST.96 Schema and in IPOs’ implementation schema; and used in an XML instance:

`st96Version` of ST.96 version 2.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V2_0">
      <xsd:attribute name="st96Version" type="xsd:token" fixed="V2_0">
            …
      </xsd:attribute>
</xsd:schema>
```

`ipoVersion` of ST.96 version 2.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V2_0">
      <xsd:attribute name="ipoVersion" type="xsd:token">
            …
      </xsd:attribute>
</xsd:schema>
```

`ApplicationBodyType` of ST.96 version 2.0:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace=”http://www.wipo.int/standards/XMLSchema/ST96/Patent”
xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common”
xmlns:pat="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V2_0">
<xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/st96Version.xsd"/>
<xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ipoVersion.xsd"/>
…………
<xsd:complexType name="ApplicationBodyType">
      …………
      <xsd:attribute ref="com:st96Version" use="required" />
<xsd:attribute ref="com:ipoVersion"/>
      …………
</xsd:complexType>
</xsd:schema>
```

`ipoVersion` and `ApplicationBodyType` of USPTO’s implementation schema version 1.0, based on the ST.96 `ApplicationBodyType` version 2.0 above:

```
<?xml version="1.0" encoding="UTF-8"?>
…
      <xsd:attribute name="ipoVersion" type="xsd:token” fixed="US_V1_0">
            …
      </xsd:attribute>
…
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:uspat="urn:us:gov:doc:uspto:patent"
targetNamespace="urn:us:gov:doc:uspto:patent"
xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common”
xmlns:pat="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V2_0">
<xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/st96Version.xsd"/>
<xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ipoVersion.xsd"/>
      …………
<xsd:complexType name="ApplicationBodyType">
      …………
      <xsd:attribute ref="com:st96Version" use="required" />
<xsd:attribute ref="com:ipoVersion" />
      …………
</xsd:complexType>
</xsd:schema>
```

An XML instance based on the element `ApplicationBody` of the USPTO's implementation schema version 1.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<uspat: ApplicationBody xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common”
xmlns:pat="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
xmlns:uspat="urn:us:gov:doc:uspto:patent”
xmlns:uscom="urn:us:gov:doc:uspto:common” "
xsi:schemaLocation="urn:us:gov:doc:uspto:patent
PE2E/V1_0/USPatent/Document/ApplicationBody_V1_0.xsd" com:st96Version="V2_0"
com:ipoVersion=”US_V1_0”>
……
</uspat:ApplicationBody>
```

[SD-31] The schema for a Document component MUST declare a required attribute named `st96Version` on the root element with a fixed value that matches the ST.96 XML Schema release version in the following format: “V”<major version number>”_”<minor version number>.

[SD-63] The schema for a Document component MUST declare an optional `ipoVersion` attribute on the root element with a fixed value that matches the IPO's implementation release version in the following format: <ST.3 Code>”_” “V”<major version number>”_”<minor version number>, for example, “US_V2_0”.

2.7 Transformability with other WIPO XML Standards

43. Before adoption of WIPO Standard ST.96, WIPO Standards ST.36, ST.66 and ST.86 had already been used by IPOs therefore, it is one of the 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.

44. 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*.

45. 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 “Unknown”, meaning that the information was absent or incorrect in the source data (XML instances conforming to ST.36, ST.66 and ST.86). Due to the historical reason, the value "Undefined" is used instead of "Unknown" in some enumeration lists.

[SD-32] ST.96 Schemas SHOULD be developed with the same 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 “Unknown” unless they are needed.

2.8 Industry-standard schemas

46. Where appropriate to the content of a document, that is, where the content is not unique to the industrial property domain, industry-standard schemas should be used. Industry-standard schemas recommended in this document are OASIS XML table schema and MathML.

47. The OASIS Table Schema, available at: http://www.oasis-open.org/docbook/xmlschema/1.0b1/calstbl.xsd, is a model schema which needs to be further specified to meet the business need. OASISTable.xsd is defined in ST.96 XML Schema on the basis of the OASIS Table schema.

48. For mathematical formulas, it is recommended to use MathML version 3. Further information on MathML version3 is available at: http://www.w3.org/TR/MathML3.

49. At the date of drafting this document there is neither a W3C recommendation nor a widely agreed XML industry standard for chemical formulae or structures. 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 specific XML standard for chemical data.

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

[SD-35] OASISTable.xsd, based on the OASIS Exchange Table schema, SHOULD be used for tables.

[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

50. 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

51. 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.

3.1.1.1 W3C built-in datatypes

52. 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

53. One of the advantages of XML schemas is the ability to create user-defined 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] Code lists SHOULD be declared as an enumeration list in a simple type.

[SD-39] A code list MAY be declared as a union of simple types.

[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] Schemas MUST declare elements and attributes for date and time values using W3C 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 enumerations MUST be restricted to the following set: ‘a-z, A-Z, 0-9, space “ “ and underscore “_”’. Enumeration values SHOULD NOT start with a numeric character.

[SD-46] Enumeration values 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

54. 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

55. 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

56. 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 only use attributes to define non-business data. For example, it is permissible to use an attribute named `sequenceNumber` for sequence numbers.

3.2.2 Elements

3.2.2.1 Cardinality of elements

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

58. 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] An occurrence indicator SHOULD NOT be used to specify a restriction that is the default within a schema. For example, `minOccurs="1"` and `maxOccurs="1"` SHOULD NOT be used but `minOccurs="2"` and `maxOccurs="3"` are acceptable.

3.2.2.2 Empty elements

59. An empty element is an element with no text 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.

60. 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

61. 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: attributes cannot be extended as elements can.

62. 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] The `use` attribute in an attribute usage SHOULD be omitted when the referenced attribute is optional since `use="optional"` is the default.

3.2.4 Element and attribute grouping

63. 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`.

64. 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.

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

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

67. 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

68. 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

69. Extension of complex types does not affect data harmonization while extension of simple types does because added components will not be exchanged. However, deriving simple types by using `union` or `list` impairs 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

70. In the context of data harmonization, restriction of complex types modifies the structure and can lead to compatibility problems blocking data exchange. In the case of simple types, restriction modifies values; no problems can be expected when exchanging data.

3.3.3 Substitution groups

71. 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

72. Like any storage system, an XML document needs to provide ways to identify and refer to pieces of the information it contains. Two features that allow XML to do so with W3C XML Schema are `xsd:key/unique/keyref` and `xsd:ID/IDREF/IDREFS`.

73. For references within the current XML document, the `xsd:ID/IDREF/IDREFS` MAY co-exist with `xsd:key/unique/keyref` in the same XML instance. The xsd:key/unique/keyref SHOULD be used for each component within a scope where uniqueness has to stay entirely within that scope. The `xsd:ID, xsd:IDREF` and `xsd:IDREFS` SHOULD be used in the XML instances where uniqueness and reference to uniqueness is required for the entire document. The `id, idref` and `idrefs` attributes are defined in Common namespace by `xsd:ID, xsd:IDREF` and `xsd:IDREFS`, respectively. The `idrefs` attribute SHOULD be used when multiple references are expected, e.g., in Claims.

74. For references that are external to the current XML document, the `extRef` attribute defined as `xsd:token` in Common namespace SHOULD be used, for example Cross Reference.

[SD-56] Schemas SHOULD use `xsd:key/xsd:keyref/xsd:unique` and/or `xsd:ID/xsd:IDREF/xsd:IDREFS` to identify constraints as appropriate.

[SD-57] Schemas SHOULD use xsd:`ID/xsd:IDREF/xsd:IDREFS` for a single reference or multiple references within the current XML document to identify constraints as appropriate.

[SD-62] Schemas SHOULD use `xsd:key/xsd:unique/xsd:keyref` within a scope where uniqueness has to stay entirely within that scope.

3.5 Schema documentation

75. The W3C schema specification allows for documentation to be included in schemas. This allows the schemas to contain embedded documentary information of each construct. Each defined or declared schema construct MUST use the `xsd:documentation` element for required documentation and, if needed, the `xsd:appInfo` element for application information within the `xsd:annotation` element. The `xsd:appInfo` element communicates 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. In general, the `xsd:documentation` element is used for human readable material and the `xsd:appinfo` element is used to provide information for tools, style sheets and other applications.

3.5.1 Schema construct documentation

76. 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.

77. 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 `xsd: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.

3.5.2 Schema header documentation

78. 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. Schema header documentation should be included in all document level elements and some aggregate and basic level components such as WIPOST3Code. In order to facilitate processing information in the header by applications, the `xsd:appInfo` element should be used for schema header documentation.

79. Table 1 lists the items that are included in the header section of Schemas.

Table 1. Header Documentation Information

Header item	Description
Created date	Date of creation of the schema (mandatory)
Last modified date	Date of last modification for the schema (mandatory)
Contact Point	Focal point to contact with questions about the schema (mandatory)
Release Notes URL	A URL where the release notes for the schema are published (optional)

The following example shows corresponding schema elements to the header items listed in the table above:

```
<xsd:annotation>
<xsd:appinfo>  <com:SchemaCreatedDate>2011-05-28</com: SchemaCreatedDate>
    <com:SchemaLastModifiedDate>2011-05-28</com: SchemaLastModifiedDate>
    <com: SchemaContactPoint>xml.standards@wipo.int</com: SchemaContactPoint >
    <com:
SchemaReleaseNoteURL>http://www.wipo.int/standards/XMLSchema/ST96/V2_0/ReleaseN
otes.pdf</com:SchemaReleaseNoteURL>
</xsd:appinfo></xsd:annotation>
```

80. 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] Document level schemas MUST include schema header documentation containing references to the created date, last modified date and contact point as mandatory and optionally a URL where the release notes of the schema are published.

4. INSTANCE DESIGN RULES

81. 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

82. 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.

83. 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.

84. 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/ST96/Common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wipo.int/standards/XMLSchema/ST96/Common
../V2_0/Common/Contact.xsd">
…….
</com:Contact>
```

85. 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/ST96/Common” 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 associate the target namespace with the location of the schema file 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

86. 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.

87. 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.

88. 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. For this reason 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`.

89. 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/ST96/Patent"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified"
attributeFormDefault="qualified">
………
</xsd:schema>
```

90. 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

91. 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/ST96/Common">
…
</com:Contact>
```

92. 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

93. 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.

94. 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

95. 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.

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

97. 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 made 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., viruses.

[ID-08] External image files 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 - LIST 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]	Schemas MUST use the ISO/IEC 10646 – UCS – Unicode 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. The only permitted exceptions are the acronyms, abbreviations and other word truncations listed in Appendix C.
[GD-05]	Type, element and attribute names SHOULD consist only of nouns, adjectives, and verbs in the present tense with the exception of acronyms, abbreviations and other word truncations listed in Appendix C.
[GD-06]	The characters used in type, element and attribute names MUST be contained in the following set: ‘a-z, A-Z and 0-9’.
[GD-07]	The maximum length of a component name SHOULD be no more than 35 characters.
[GD-08]	Type, element and attribute names SHOULD be concise and self-explanatory.
[GD-09]	Element names MUST use the upper camel case (UCC) convention. For example, `CountryCode`.
[GD-10]	Type names MUST use the UCC convention and have the suffix Type. For example, `ApplicantType`.
[GD-11]	Attribute names MUST use the 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 in all lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case as listed in Appendix C.
[GD-14]	Acronyms and abbreviations MUST appear as listed in Appendix C for element and type names.
[GD-15]	Complex type names SHOULD include a meaningful Object Class Term.
[GD-16]	When we need two Object Class terms in a given component name due to the nature of business entity, association complex type can be used. Names of an association complex type SHOULD use a structure of Object Class term of the associating complex type, a Property term that describes the nature of the association, and the Object Class of the associated complex type. Qualifiers may precede the Object Class and Property Term. For example, in `ApplicantResidenceAddress, Applicant` is Object Class of associating complex type, `Residence` is the Property, and `Address` is the Object Class of associated complex type.
[GD-17]	Names of basic elements (those based on a simple or complex type that do not permit children) SHOULD consist of an Object Class Term, Property Term, and where applicable a Representation Term. Qualifier Terms may precede the Object Class term and Property 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]	When a name contains an Object Class Term, a Property Term, and a Representation Term, the Object Class Term MUST precede the Property Term and the Property Term MUST precede the Representation Term. A Qualifier Term SHOULD precede the associated Object Class Term or Property Term.

Rule ID	Rule
[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]	Where a representation term is required, the Representation Terms in Appendix B MUST be used for representation terms in Basic component names.
[GD-24]	Within a namespace, all type, element and attribute names 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 an element or a type which contains a collection of contextually related components SHOULD have the “Bag” suffix. For example, `EmailAddressBag` represents a collection of `EmailAddress` elements.
[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.
[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 filenames MUST belong to the following set: ‘a-z, A-Z, 0-9, underscore “_”, and period “.”’.
[GD-31]	A Schema filename MUST consist of two compulsory parts with one delimiter and optional version information with two additional delimiters, i.e.: <component name>{“_”“V”<major version number>“_”<minor version number>“}.”<file extension>;. For example, EmailAddressType.xsd, languageCode.xsd, ApplicationBody_V1_0.xsd.
[GD-32]	A draft Schema filename MUST consist of four compulsory parts with two delimiters and optional version information with two additional delimiters, i.e.: <component name>{“_”“V”<major version number>“_”<minor version number>“}_”“D”<revision number>“.”<file extension>, for example, Contact_D3.xsd, TrademarkApplication_V1_1_D1. If a draft schema is based on an existing schema and has version information in its filename, the major and minor version numbers in the draft schema filename SHOULD be the same as specified in the schema file that the draft schema is based on. If a draft schema is new, the major version number in the draft schema filename SHOULD be the same number as specified in the corresponding namespace and a minor version number in the draft schema file SHOULD be zero “0”.

Schema Design Rules

Rule ID	Rule
[SD-01]	Patent Component schema modules, Trademark Component schema modules, and Design Component schema modules MUST use the `xsd:import` construct to reference Common Component schema modules.
[SD-02]	Common Component schema modules MUST NOT refer to Patent Component schema modules, Trademark Component schema modules or Design Component schema modules.
[SD-03]	A Patent Component, Trademark Component or Design Component schema module MAY make reference to Common Component schema modules, schemas in the same Component module, and approved industry-standard schemas, but MUST NOT refer to other Component schema modules. For example, Patent schemas MUST NOT refer to Trademark schemas and vice versa.
[SD-04]	Existing schemas MUST be used wherever applicable prior to creating new schemas.
[SD-05]	Schemas SHOULD use elements and types defined in existing schemas to the maximum extent possible.
[SD-06]	All types, elements and attributes MUST be globally declared.
[SD-07]	Schemas MUST NOT use `xsd:redefine` construct.
[SD-08]	Schemas MUST use namespaces.
[SD-09]	Published namespace declarations SHOULD NOT be changed.
[SD-10]	Schemas MUST declare the W3C Schema namespace.

Rule ID	Rule
[SD-11]	Schemas MUST use namespace qualifications for all W3C Schema constructs.
[SD-12]	Schemas MUST use the URI-formatted namespace.
[SD-13]	All schemas MUST have attributes `elementFormDefault` and `attributeFormDefault` with value “`qualified`” in the root `xsd:schema` element.
[SD-14]	The names for namespaces MUST have the following structure: `http://www.wipo.int/standards/XMLSchema/ST96/<category>`; where <category> is a token identifying the domain of Schema being used: `Common`, `Patent`, `Trademark`, or `Design`.
[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 the target namespace using the `xsd:targetNamespace` attribute.
[SD-17]	The schema’s `targetNamespace` MUST match the namespace name of one of the declared namespaces, but not the W3C namespace.
[SD-18]	Common Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Common as the target namespace.
[SD-19]	Patent Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Patent as the target namespace.
[SD-20]	Trademark Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Trademark as the target namespace.
[SD-21]	Design Component Schemas MUST use http://www.wipo.int/standards/XMLSchema/ST96/Design as the target namespace.
[SD-22]	Schemas SHOULD NOT use default namespace.
[SD-23]	New minor versions of Schemas MUST be able to validate all instance documents created with previous minor versions of that Schema with the same major version.
[SD-24]	The Schema major version MUST be incremented if the new schema is not able to validate existing XML instance documents created with the current major version.
[SD-25]	*Deleted*
[SD-26]	*Deleted*
[SD-27]	The major and/or minor version number of a schema SHOULD be changed when an included or imported schema is updated.
[SD-28]	When any schema construct is altered, all schemas in the release MUST undergo the same version number 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 consist of both major version number and a minor version number for each schema file in the following format: “V”<major version number>”_”<minor version number>.
[SD-31]	The schema for a Document component MUST declare a required attribute named `st96Version` on the root element with a fixed value that matches the ST.96 XML Schema release version in the following format: “V”<major version number>”_”<minor version number>.
[SD-32]	ST.96 Schemas SHOULD be developed with the same 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 “Unknown” unless they are needed.
[SD-34]	MathML, version 3, SHOULD be used for mathematical formulas.
[SD-35]	OASISTable.xsd, based on the OASIS Exchange Table schema, SHOULD be used for tables.

Rule ID	Rule
[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]	Code lists SHOULD be declared as an enumeration list in a simple type.
[SD-39]	A code list MAY be declared as a union of simple types.
[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]	Schemas MUST declare elements and attributes for date and time values using W3C 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 enumerations MUST be restricted to the following set: ‘a-z, A-Z, 0-9, space “ “ and underscore “_”’. Enumeration values SHOULD NOT start with a numeric character.
[SD-46]	Enumeration values 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 only use attributes to define non-business data. For example, it is permissible to use an attribute named `sequenceNumber` for sequence numbers.
[SD-49]	An occurrence indicator SHOULD NOT be used to specify a restriction that is the default within a schema. For example, `minOccurs="1"` and `maxOccurs="1"` SHOULD NOT be used but `minOccurs="2"` and `maxOccurs="3"` are acceptable.
[SD-50]	Empty elements MUST NOT be defined in schemas except for line break.
[SD-51]	The `use` attribute in an attribute usage SHOULD be omitted when the referenced attribute is optional since `use="optional"` is the default.
[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` and/or `xsd:ID/xsd:IDREF/xsd:IDREFS` to identify constraints as appropriate.
[SD-57]	Schemas SHOULD use `xsd:ID/xsd:IDREF/xsd:IDREFS` for a single reference or multiple references within the current XML document to identify constraints as appropriate.
[SD-58]	All schemas SHOULD include schema construct documentation using the `xsd: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]	Document level schemas MUST include schema header documentation containing references to the created date, last modified date and contact point as mandatory and optionally a URL where the release notes of the schema are published.
[SD-62]	Schemas SHOULD use `xsd:key/xsd:unique/xsd:keyref` within a scope where uniqueness has to stay entirely within that scope.

Rule ID	Rule
[SD-63]	The schema for a Document component MUST declare an optional `ipoVersion` attribute on the root element with a fixed value that matches the IPO's implementation release version in the following format: <ST.3 Code>"_" "V"<major version number>"_"<minor version number>, for example, "US_V2_0".
[SD-64]	Schema release folder, a document level schema filename and a flattened schema filename MUST contain matching version information comprising the major and minor version number.

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 associate the target namespace with the location of the schema file 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 image files 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 or free format.	`xsd:token`
Date	The notion of a specific point in time, expressed by year, month, and day.	`DateType` or `xsd:date`
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 string of numeral or alphanumeric characters expressing label, value, quantity or identification.	`xsd:positiveInteger`, `xsd:string` or `xsd:token`
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. `Quantity` is used to represent a counted number of things. `Quantity` should be used for simple properties of an object seen as a composite or collection or container to quantify or count its components. `Quantity` should always express a counted number of things, and the property will be such as total, shipped, loaded, stored. `QuantityType` should be used for components that require unit information; and `xsd:nonNegativeInteger` should be used for countable components which do not need unit information.	`QuantityType` or `xsd:nonNegativeInteger`
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`, `LocalizedTextType` or `OrderedTextType`
Time	A designation of a specified chronological point within a period.	`xsd:time`
DateTime	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

For abbreviations, in principle, the first character of the word or phrase should be capital and last characters be lower case (e.g., "Pre", "BioDeposit") except abbreviations which are already well-known, e.g., "IDREF", "W3C", "ST3". In the case of acronyms, all characters should be capital, e.g., "WIPO".

Alt	Alternate text for image
B	Bold
BioDeposit	Biological Deposit
Br	Break
CDX	CambridgeSoft proprietary ChemDraw file format
CPC	Cooperative Patent Classification
DD	Definition Description
Del	Deleted text
DL	Definition List
DOI	Digital Object Identifier
DT	Definition Term
DTD	Document Type Definition
DWF	Design Web Format
DWG	Drawing
ECLA	European Classification
ExtRef	References that are external to the current XML document
H<n>	The “n” indicates the level of Heading with a specific value from 1 to 15 digit number. It means, in the enumeration value, this abbreviation represents one of H1 to H15. For example, “H1” means “Heading 1”.
I	Italic
ID	Identifier for system identification
IDREF	Identifier Reference
IDREFS	Identifier References
IGES	Initial Graphic Exchange Specification
Ins	Inserted text
IP	Industrial Property
IPC	International Patent Classification
IPCR	International Patent Classification Reform
IPO	Industrial Property Office
IPR	Industrial Property Right
ISO	International Organization for Standardization
LCC	Lower Camel Case
LI	List Item
MPEG	Moving Picture Experts Group
MOL	File format for holding information about the atoms, bonds, connectivity and coordinates of a molecule
NB	File format for Mathematica notebooks
NPL	Non Patent Literature
O	Over score

OASIS	Organization for the Advancement of Structured Information Standards
OCR	Optical character recognition
OL	Ordered List
P	Paragraph
PAN	Primary Account Number
PCT	Patent Cooperation Treaty
PKCS7	In cryptography, PKCS is a group of public-key cryptography standards and PKCS #7 (PKCS7) is for the Cryptographic Message Syntax Standard which describes general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes.
Pre	Preformatted text
S	Strike through text
ST3	WIPO Standard ST.3
ST13	WIPO Standard ST.13
Sub	Subscript
Sup	Superscript
SVG	Scalable Vector Graphics image
SWF	Small Web Format
SWIFT	Society for Worldwide Interbank Financial Telecommunication
ThreeDM	Dimensional Modeling
ThreeDS	3D Studio
U	Underlined
UCC	Upper Camel Case
UL	Unordered List
UPOV	The International Union for the Protection of New Varieties of Plants
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WIPO	World Intellectual Property Organization
WMV	Windows Media Video
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.13: Numbering of applications for IPRs
- 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]