R

WIPO
ВСЕМИРНАЯ ОРГАНИЗАЦИЯ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

CWS/10/6 REV.
ОРИГИНАЛ: АНГЛИЙСКИЙ
ДАТА: 10 НОЯБРЯ 2022 ГОДА

Комитет по стандартам ВОИС (КСВ)

Десятая сессия
Женева, 21–25 ноября 2022 года

ПРЕДЛОЖЕНИЕ В ОТНОШЕНИИ НОВОГО СТАНДАРТА ВОИС НА JSON

Документ подготовлен Международным бюро

РЕЗЮМЕ

1. От имени Целевой группы по XML для ПС Международное бюро представляет окончательный проект нового стандарта ВОИС по формату JavaScript Object Notation (JSON) для рассмотрения на десятой сессии Комитета по стандартам ВОИС (КСВ). Окончательный проект включает ряд правил и условных обозначений, а также набор схем JSON, основанных на XML-схемах, соответствующих стандарту ВОИС ST.96, и примеров в формате JSON.

ИСТОРИЯ ВОПРОСА

2. На своей пятой сессии, состоявшейся в 2017 году, КСВ одобрил включение новой задачи № 56, чтобы обеспечить основу для начала работы Целевой группы по XML для ПС над новым стандартом ВОИС, который содержит набор рекомендаций в отношении обработки и передачи данных об интеллектуальной собственности (ИС) с использованием интерфейсов прикладного программирования (API). В настоящем описании задачи указано, что JSON может использоваться в качестве формата полезной нагрузки.

3. На своей седьмой сессии КСВ сформулировал новую задачу № 64, описание которой приводится ниже, и поручил выполнение этой задачи Целевой группе по XML для ПС (см. пункты 58–59 документа CWS/7/29):

> «Подготовить предложение о рекомендациях для ресурсов в формате JavaScript Object Notation (JSON), соответствующих стандарту ВОИС ST.96, для

использования в целях подачи, обработки, публикации и/или обмена информацией в области интеллектуальной собственности».

4. На своей восьмой сессии, состоявшейся в 2020 году, КСВ одобрил новый стандарт ВОИС ST.90 «Рекомендация по обработке и передаче данных об интеллектуальной собственности с использованием API (интерфейсов программирования приложений) для веб-сервисов». Стандарт ВОИС ST.90 включает примеры в форматах XML и JSON. В то время как стандарт ВОИС ST.90 ссылается на стандарт ВОИС ST.96 в качестве эталона для XML-схем, для формата JSON не существует эталона, поскольку в то время не существовало стандарта ВОИС для JSON. Стандарт ВОИС ST.90 содержит сноску, в которой говорится: «Стандарт ВОИС на JSON в настоящее время обсуждается, но он будет основан на стандарте ВОИС ST.96».

5. На своей девятой сессии, состоявшейся в 2021 году, КСВ отметил, что окончательный проект нового стандарта на JSON будет готов для рассмотрения и принятия на десятой сессии (см. пункт 20 документа CWS/9/25). Со времени проведения девятой сессии КСВ в рамках Задачи № 64 Целевой группе по XML для ПС удалось завершить работу над проектом стандарта после серии обсуждений этого вопроса в ходе заседаний, по электронной почте и на форуме wiki.

ПРЕДЛАГАЕМЫЙ НОВЫЙ СТАНДАРТ ВОИС

6. Формат JSON постепенно внедряется и используется ведомствами интеллектуальной собственности (ВИС) и отраслью ИС, в то время как XML (расширяемый язык разметки), основанный на стандартах ВОИС в отношении XML, по-прежнему используется достаточно широко. Стандарт ВОИС ST.96 содержит рекомендацию в отношении того, как следует обращаться с XML-ресурсами при подаче, обработке и обмене информацией по различным видам ИС, т. е. патентам, товарным знакам, промышленным образцам, географическим указаниям и авторскому праву. Стандарт ВОИС ST.96 внедряется ВИС в том виде, в каком он опубликован или изменен по мере необходимости.

7. Целевая группа по XML для ПС разработала проект стандарта на JSON с учетом того, что необходимо было обеспечить согласованность и совместимость данных между форматами XML и JSON для облегчения обмена данными между ВИС и распространения данных ВИС в этих двух форматах. Совместимость и согласованность данных могут быть достигнуты при помощи совместимых схем XML и схем JSON, которые будут использоваться для проверки данных в XML и JSON соответственно.

8. На момент подготовки настоящего документа еще не существовало международного стандарта на JSON, одобренного отраслью. Существуют только предварительные спецификации, а формат JSON продолжает развиваться. Проект 2020-12 — это последняя версия проекта спецификации схемы JSON, а версия 5.0 — это последняя версия стандарта ВОИС ST.96. Таким образом, предлагаемый стандарт на JSON основан на этом проекте спецификации схемы JSON и версии 5.0 стандарта ВОИС ST.96. Следует отметить, что новая версия ST.96 6.0 будет опубликована в октябре 2022 года.

Цели

9. Настоящий предлагаемый стандарт содержит ряд рекомендаций в отношении представления данных об интеллектуальной собственности в формате JSON. Основные цели этого стандарта заключаются в ~~обеспечении следующих преимуществ~~ следующем:

– дать рекомендации по разработке и совершенствованию передовой практики работы с данными об ИС в формате JSON ~~дать рекомендации по разметке данных в формате JSON~~;

- обеспечить согласованность путем использования схем и примеров в формате JSON на основе стандарта ВОИС ST.96 для обмена данными об ИС;
- рекомендовать принципы проектирования для расширения предоставленных схем JSON или создания новых совместимых схем JSON; а также
- повысить эффективность обмена данными, рекомендуя повторное использование ресурсов в формате JSON среди ВИС, а также данных, предоставляемых общественности.

Сфера применения

10. Этот предлагаемый стандарт призван служить рекомендацией для ВИС и других организаций, которые генерируют данные об ИС или вносят в них изменения в качестве ресурсов в формате JSON. Соблюдение этого стандарта требуется для обмена данными между ВИС с использованием ресурсов в формате JSON, таких как схемы, примеры, сообщения и файлы полезной нагрузки, ~~в качестве формата полезной нагрузки~~ для API.

11. Структура предлагаемого стандарта выглядит следующим образом:

- **Основная часть**: определение общих правил разработки, схемы JSON и правил построения схемы, идентификаторов схемы JSON и правил составления примера в формате JSON;
- **Приложение I:** правила преобразования XML-схем, соответствующих стандарту ST.96, в схемы JSON, содержащее дополнение «Инструмент преобразования XSD ST.96 в схемы JSON»;
- **Приложение II:** схемы JSON, преобразованные из XML-схем, соответствующих версии 5.0 стандарта ВОИС ST.96;
- **Приложение III:** типовые примеры в формате JSON, соответствующие типовым XML примерам, представленным в приложении VII стандарта ВОИС ST.96;
- **Приложение IV:** сокращения и аббревиатуры на основе ВОИС ST.96; а также
- **Приложение V:** условия репрезентации, основанные на ВОИС ST.96.

12. Международное бюро предлагает следующее название для нового стандарта ВОИС:

«*Стандарт ВОИС ST.97: Рекомендации в отношении данных об интеллектуальной собственности с использованием JSON (JavaScript Object Notation)»*

13. КСВ следует принять к сведению, что предлагаемый стандарт на JSON не касается архитектуры программного обеспечения или языков реализации. Предлагаемый стандарт, включая все перечисленные выше приложения, воспроизводится в качестве приложения к настоящему документу.

Дальнейшая работа

14. При подготовке предлагаемого стандарта на JSON рабочая группа по XML для ПС выявила и решила множество проблем. Однако остаются нерешенными некоторые проблемы, которые необходимо решить, а также внести изменения, необходимые в связи с развитием спецификации схемы JSON, в том числе:

- упрощение структуры схем JSON в соответствии с отраслевой практикой использования JSON, например, за счет удаления ненужных вложений из схемы

JSON, являющихся результатом преобразования из XML-схем, соответствующих ST.96;

- усовершенствование структуры данных для более точной проверки данных, для чего потребуется проведение анализа бизнес-экспертами;
- обновление набора схем JSON в соответствии с изменениями, внесенными в версию 6.0 ST.96, которая ~~будет~~ была опубликована в октябре 2022 года;
- добавление правил преобразования примеров XML в JSON и соответствующих инструментов; а также
- при необходимости пересмотр нового стандарта с учетом любых изменений в спецификации схемы JSON либо в следующем проекте, либо в официальной публикации.

ПЕРЕСМОТРЕННОЕ ОПИСАНИЕ ЗАДАЧИ № 64

15. Как только КСВ примет предлагаемый новый стандарт на JSON, задача № 64 будет считаться выполненной, и Целевая группа XML для ПС успешно завершит работу над этой задачей. Однако, как указано выше, этот стандарт следует сохранить для будущих пересмотров. В связи с этим предлагается пересмотреть описание Задания № 64 и сформулировать его следующим образом:

«*Обеспечить необходимый пересмотр и обновление стандарта ВОИС ST.97*».

ПОДДЕРЖАНИЕ НОВОГО СТАНДАРТА ВОИС

16. Хотя Целевая группа по XML для ПС успешно выполнила задачу № 64 и подготовила эту первую версию стандарта ВОИС на JSON на основе стандарта ВОИС ST.96 на XML, Международное бюро предлагает поручить Целевой группе по API решение пересмотренной задачи № 64, поскольку она также управляет стандартом ВОИС ST.90. Это связано с тем, что JSON так часто используется в качестве формата полезной нагрузки для API архитектуры RESTful.

17. Принимая во внимание постоянный пересмотр и обновление стандарта ВОИС ST.96, КСВ установил «ускоренную» процедуру рассмотрения и/или принятия пересмотренных версий этого стандарта Целевой группой по XML для ПС. Поскольку ожидается, что новый стандарт на JSON будет постоянно пересматриваться наряду с внесением изменений в стандарт ВОИС ST.96 и развитием спецификации схемы JSON, предлагается установить еще одну «ускоренную» процедуру рассмотрения и/или принятия пересмотренных версий нового стандарта JSON в следующем порядке:

(a) любое предложение о пересмотре стандарта ВОИС ST.97 представляется Целевой группе напрямую или через Секретариат для рассмотрения и одобрения;

(b) назначенная Целевая группа временно уполномочена принимать изменения к стандарту ВОИС ST.97;

(c) если предложение о пересмотре стандарта ВОИС ST.97 вызовет споры, то оно будет представлено на рассмотрение КСВ, например если назначенным членам Целевой группы не удается достичь консенсуса; а также

(d) назначенный руководитель Целевой группы информирует КСВ о любом пересмотре стандарта ВОИС ST.97, принятом Целевой группой, на следующей сессии КСВ.

РЕДАКЦИОННЫЕ ПОПРАВКИ К СТАНДАРТУ ВОИС ST.90

18. Поскольку на этой сессии Целевая группа по API не представила ни одного документа, Международное бюро отметило несколько редакционных поправок, которые

необходимо внести в стандарт ВОИС ST.90 в случае принятия предложенного стандарта ST.97. На рассмотрение КСВ представлены следующие предлагаемые редакционные поправки:

– добавление ссылки на новый стандарт ВОИС ST.97 в раздел «Ссылки»;

– внесение изменения в пункт 33 стандарта ВОИС ST.90 путем включения ссылки на стандарт ВОИС ST.97 в виде подчеркнутой фразы, чтобы этот пункт читался следующим образом:

  «API должны поддерживать запросы и ответы XML и JSON. Для XML ответы должны соответствовать стандартам ВОИС с использованием XML, таким как ST.96, а <u>для JSON ответы должны соответствовать стандарту ВОИС ST.97</u>. Следует использовать согласованное сопоставление между этими двумя форматами»; а также

– удаление сноски 7 к пункту 33 стандарта ST.90, которая воспроизводится ниже:

  «Спецификация JSON и схема JSON, основанные на ST.96, в настоящее время обсуждаются Целевой группой по XML для ПС, для того чтобы представить их на рассмотрение на 8-й сессии КСВ в ноябре 2020 года для рассмотрения/принятия в качестве нового стандарта ВОИС. Между тем, этот стандарт рекомендует использовать конвенцию BadgerFish из-за его простоты, пока не будет предоставлена схема JSON. Некоторые ВИС, такие как ЕПВ, также ссылаются на него, www.epo.org/searching-for-patents/data/web-services/ops.html».

19. Предлагаемые редакционные поправки в пункте 18 выше не являются исчерпывающими. Поэтому предлагается, чтобы Секретариат, если потребуется, внес необходимые редакционные поправки перед повторной публикацией пересмотренной новой редакции стандарта ВОИС ST.90.

20. КСВ предлагается:

(a) принять к сведению информацию, содержащуюся в настоящем документе и приложении к нему (окончательный проект стандарта на JSON);

(b) рассмотреть и одобрить название предлагаемого стандарта «Стандарт ВОИС ST.97: Рекомендации в отношении данных об интеллектуальной собственности с использованием JSON», как указано в пункте 12;

(c) рассмотреть и одобрить новый стандарт ВОИС ST.97, приведенный в приложении к настоящему документу;

(d) рассмотреть и одобрить измененное описание задачи № 64, приведенное в пункте 15;

(e) поручить Целевой группе по API решение задачи № 64, как указано в пункте 16;

(f) рассмотреть и одобрить ускоренную процедуру пересмотра стандарта ВОИС ST.97, как указано в пункте 17; и

(g) одобрить внесение изменений в стандарт ВОИС ST.90, который должен содержать ссылку на одобренный стандарт ST.97, и поручить Секретариату внести другие редакционные поправки и опубликовать пересмотренный стандарт ST.90, как указано в пунктах 18 и 19.

[Приложение (проект стандарта на JSON) следует]

WIPO STANDARD ST.XX

RECOMMENDATION FOR PROCESSING OF INTELLECTUAL PROPERTY DATA USING JSON

Proposal presented by the XML4IP Task Force for consideration at the tenth session of the Committee on WIPO Standards

Editorial note: To support understanding of certain rules, notes are provided throughout the proposed draft standard but will be removed from the document after adoption of the Standard, upon publication

TABLE OF CONTENTS

WIPO STANDARD ST.XX....1
TABLE OF CONTENTS....1
TABLE OF CONTENTS (TABLES)....2
1.INTRODUCTION....3
2.DEFINITIONS AND TERMINOLOGY....3
3.GENERAL NOTATIONS....3
3.1 Rule Identifiers....3
3.2 Sample JSON Data Structure....4
4.SCOPE....4
5.JSON GENERAL DESIGN RULES....4
5.1 Overview....4
5.2 JSON Naming Conventions....4
6.JSON SCHEMA DESIGN RULES....6
6.1 Overview....6
6.2 Modularity....6
6.3 Documentation....14
6.4 Filename....15
6.5 JSON Schema Properties Structuring....15
7.JSON SCHEMA CONSTRUCTS DESIGN RULES....16
7.1 Overview....16
7.2 Properties....16
7.3 Definitions....16
7.4 Type Definitions....17
7.5 JSON Primitive Type....17
7.6 Code Lists....17
7.7 Arrays....18
7.8 Objects....18
8JSON SCHEMA IDENTIFIERS....18
8.1 Overview....18
9. JSON INSTANCE DESIGN RULES....18
9.1 Order of Properties....18
9.2 JSON instance validation....19
8.REFERENCES....19
WIPO Standards....19
Standards and Industry Specifications....19
ST.XX - ANNEX I....20
TRANSFORMATION RULES FROM ST.96 XSD TO JSON SCHEMA AND GUIDELINES FOR USE....20
INTRODUCTION....20
TRANSFORMATION ALGORITHM AND GUIDELINES....20
SCOPE....20
FILENAME TRANSFORMATION....20
NAMESPACES TRANSFORMATION....20
BUILT-IN DATA TYPES TRANSFORMATION....21
XSD DEFINITION....22
GLOBAL SCHEMA REFERENCING....28
XSD COMPOSITORS....29
ELEMENTS....34
ATTRIBUTES....35
SIMPLETYPE....36
COMPLEXTYPE....36
Simple Content....36

Complex Content *36*
Mixed Content *36*
ANNOTATION 38
xsd:appinfo *38*
xsd:documentation *39*
UNION 39
EXTENSION 40
RESTRICTION 41
ENUMERATION 41
CONSTRAINING FACETS 41
Pattern *43*
GROUP 43
TRANSFORMATION OF EXTERNAL XSD DEPENDENCIES 44
APPENDIX 45
Requirements *45*
Usage *45*
Download Executable Jar *45*
ANNEX II 46
JSON SCHEMA 46
ANNEX III 47
JSON EXAMPLE INSTANCES 47
ANNEX IV 48
LIST OF ACRONYMS AND ABBREVIATIONS 48
ANNEX V 50
REPRESENTATION TERMS 50

TABLE OF CONTENTS (TABLES)

Table 1. JSON Schema header documentation items 14
Table 2: Transformation of simple XSD data types 21
Table 3 Transformation of XSD constraining facets (X is the constraint numeral value) 42

1. INTRODUCTION

This Standard provides recommendations for designing, creating or updating JavaScript Object Notation (JSON) resources for use in filing, processing, exchanging or publishing all types of Intellectual Property (IP) data. This Standard also considers rules to transform WIPO Standard ST.96 eXtensible Markup Language (XML) Schemas (XSDs) to JSON Schemas that meet the aforementioned recommendations.

This Standard is intended to:

- Provide guidance on designing and developing IP JSON data best practices~~data mark-up in JSON format~~;
  - Ensure consistency by providing JSON Schemas and Instances based on WIPO Standard ST.96 for exchanging IP data;
  - Recommend design principles for extending the provided JSON Schemas or creating new conformant JSON Schemas; and
  - Improve data exchange efficiency by promoting reuse of JSON resources among Intellectual Property Offices (IPOs), as well as data provided to the public.

2. DEFINITIONS AND TERMINOLOGY

For purposes of the Standard, the following terminologies are used:

- The term “JSON resources” is intended to refer to any of the components used to create and operate a JSON implementation according to this Standard;
- The terms "`object`", “`object type`”, "`property`", “`member`”, "`property name`", "`property value`", “`property type`”, “`keyword`” and “`definition`” in this Standard are to be interpreted as defined in JSON Schema Core, version draft-2020-12[1];
  - The term 'construct' in this Standard should be interpreted as a 'building block' from which JSON schemas are built;
  - The term “global definition” is used for a definition that can be referenced by other definitions in the same schema or by definitions in other schemas; and
  - In this Standard, “MUST”, “MUST NOT”, "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119. Non-capitalized forms of these words are used in the regular English sense.

3. GENERAL NOTATIONS

The following notations are used throughout this Standard:

  - <>: 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; and
  - `Courier New` font: Indicates JSON keywords, JSON property names and XSD elements and attributes.

3.1 Rule Identifiers

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

  - The value “JXX” is a prefix to categorize the type of rule as follows:

    (a) JGD for general design rules;

    (b) JSD for JSON schema design rules;

    (c) JCD for construct design rules; and

[1] JSON Schema version is subject to change because it has not achieved RFC status; it has not been adopted by an IETF Working Group. This version of the Standard is based on the latest version, i.e. 2020-12, available at https://json-schema.org/draft/2020-12/json-schema-core.html

(d) JID for instance design rules

– 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 [JGD-10] identifies the tenth general design rule. The rule [JGD-10] can be placed between rules [JGD-05] and [JGD-06] instead of following [JGD-09] if that is the most appropriate location for this rule.

– The rule identifier of the deleted rule will be kept while the rule will be replaced with the text “Deleted”.

3.2 Sample JSON Data Structure

Sample JSON data structures appear within text boxes using a fixed-width font. Sample JSON data structure syntax are highlighted for easier readability.

4. SCOPE

This Standard is intended to provide JSON resources to be used for filing, publication, processing, and exchange of IP data and information. This Standard is aimed at providing guidance to IPOs and other Organizations which deal with data and documents of patent, trademark, industrial design, geographical indication and/or copyright orphan work.

This Standard aims to provide guidance to IPOs and other Organizations that create or modify IP data as JSON resources. Compliance with this Standard is required for data exchange between IPOs using JSON resources such as Schemas, instances, messages and payloads for Application Programming Interfaces (APIs).

The design rules and transformation rules are written considering the Design Rules and Conventions of WIPO Standard ST.96. ST.96 design rules and conventions are not one-to-one mapping to JSON design rules and conventions and therefore the ST.96 design rules are duplicated and in some cases are slightly modified where applicable.

This Standard includes the following Annexes:

– Annex I: Transformation Rules from ST.96 XML Schemas to JSON schemas, which contains the Appendix: a Transformation Tool which transforms ST.96 XSDs to JSON Schemas;

– Annex II: JSON Schemas which were transformed from WIPO Standard ST.96 XML schemas, version 5.0[2];

– Annex III: JSON example instances;

– Annex IV: List of acronyms and abbreviations; and

– Annex V: Representational terms.

This Standard excludes the following:

(a) Software architectural concerns; and

(b) Implementation languages.

5. JSON GENERAL DESIGN RULES

5.1 Overview

This section contains general, high-level JSON design rules and guidelines that apply to all JSON data exchange and JSON development efforts, rather than to a specific programming language marshalling/ unmarshalling data to/from JSON. The general rules and guidelines, listed below, provide the common foundation for JSON Schema, JSON instance, and JSON data structure development for all data to include IP data and non-IP data such as mixed-content data. Levels of nesting SHOULD be kept to a minimum when creating new JSON Schemas, JSON instances, and JSON data structure development that are not available in WIPO Standard ST.96 or the compatibility with ST.96 is not necessarily to be considered.

5.2 JSON Naming Conventions

These conventions are necessary to ensure consistency, uniformity, and comprehensiveness in the naming and defining of all JSON resources.

[2] The transformed JSON Schemas have the same tag names and the data structure as defined in ST.96, Annex III, including mixed contents XML components and external standards, i.e., MathML and Oasis Table, for the interoperability with data in ST.96 format.

These JSON naming conventions are based on the guidelines and principles described in document ISO 11179 Part 5 - Naming and Identification Principles. The name of objects and property names 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.

– 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.

– *Representation Term* categorizes the format of the data element into broad types. Representation Terms listed in Annex V should be used.

[JGD-01] Object type and property 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 Annex IV.

[JGD-02] Object type and property 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 Annex IV .

[JGD-03] The characters used in property names MUST be contained in the following set: ‘a-z, A-Z and 0-9’.

[JGD-04] The maximum length of object type and property names SHOULD be no more than 35 characters.

[JGD-05] Object type and property names SHOULD be concise and self-explanatory.

[JGD-06] Object type and property names MUST use the lowerCamelCase (LCC) convention. For example, `"currencyCode"`:"EUR".

[JGD-07] Object type names MUST use the LCC convention and have the suffix Type. For example, `applicantType`.

[JGD-08] The acronyms and abbreviations listed in Annex IV MUST always be used instead of the complete extended name.

[JGD-09] Acronyms and abbreviations MUST appear as listed in Annex IV for property and object type names.

[JGD-10] An Object Class Term MUST always have the same semantic meaning throughout a specific IP domain such as patents, trademarks, industrial designs, geographical indications or copyright and MAY consist of more than one word. For example, `contactInformation`.

[Notes: the word “namespace” in ST.96 has been replaced with “IP domain” in JSON]

[JGD-11] A Property Term in a name MUST be unique within the context of an Object Class but MAY be reused across different Object Classes.

[JGD-12] A Qualifier Term MAY be attached to an Object Class Term or a Property Term if necessary to make a name unique.

[JGD-13] 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. For example, `claimTotalQuantity`.

[JGD-14] If the Property Term ends with the same word as the Representation Term (or an equivalent word) then the Representation Term MUST be removed.

[JGD-15] Where a representation term is required, the Representation Terms in Annex V MUST be used for representation terms in Basic component names.

[JGD-16] Within an IP domain, all object type and property names MUST be unique.

[JGD-17] Word(s) in a name SHOULD be in singular form unless the concept itself is plural. For example, `totalMarkSeries`.

[JGD-18] The name of a property or an object type which contains a collection of contextually related components SHOULD have the “Bag” suffix. For example, `emailAddressBag` represents a collection of `emailAddress` elements.

[JGD-19] Connecting words like “and”, “of” and “the” SHOULD NOT be used in object type and property names unless they are part of the business terminology.

[JGD-20] Object type and property names MUST NOT be translated, changed or replaced for any purpose.

[JGD-21] Object type and property names MUST NOT refer to article and rule numbers. For example, `PCTRule702C` for the PCT.

[JGD-22] Levels of nesting SHOULD be kept to a minimum when creating new JSON Schemas, JSON instances, and JSON data structure/fragments that are not available in WIPO Standard ST.96.

[JGD-23] For the new Object type and property names that are not defined in WIPO Standard ST.96 or are not expected to be compatible with a component name which will be defined in ST.96, inline descriptive terms or names SHOULD be used rather than having generic or short object types or property names that are single-property objects. For example, instead of "i`nventor`": { "`fullName`": "`Thomas Edison" }`, "`inventorFullName`": "`Thomas Edison`" is preferred.

[Notes: This rule recommends to follow JSON industry practice in case if the names are not related to XML data based on current and future ST.96.]

6. JSON SCHEMA DESIGN RULES

6.1 Overview

The JSON Schema describes the structure of the JSON instance, which expresses the constraints on the structure and content of the document. This Standard should be aligned with the industry JSON schema specification. The latest version available at time of publishing this Standard is draft 2020-12 and this version of the Standard refers to this draft specification.

[JSD-01] JSON Schemas MUST conform to JSON Schema specifications: JSON Schema Core, version 2020-12, available at https://json-schema.org/latest/json-schema-core.html, and JSON Schema Validation, version 2020-12, available at https://json-schema.org/latest/json-schema-validation.html.

[JSD-02] JSON Schemas MUST indicate that they conform to version 2020-12 of JSON Schema by using the `$schema` keyword with the value "https://json-schema.org/draft/2020-12/schema".

Example: Indicating the version of JSON Schema
`"$schema": "https://json-schema.org/draft/2020-12/schema"`

The schema should be encoded using UTF-8 for maximum interoperability.

[JSD-03] JSON Schemas MUST use the ISO/IEC 10646 – UCS – Unicode character set. UTF-8 MUST be used for encoding Unicode characters.

6.2 Modularity

Modularity allows the creation of schema components to support flexibility in design and reusability. In the design, it is recommended to avoid the definition of all the properties and logical components in a single monolithic JSON Schema, which prevents the ability to share and reuse individual properties or logical components defined as a group in a schema.

Below is the schema that does **not** adhere to the modularity principle. This is NOT recommended by this Standard.

applicationNumber.json (Incorrect example compound schema document)

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "description" : "Description: Numbers used by IPOs in order to identify
each application received; Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "ipOfficeCode" : {
          "anyOf" : [ {
           "type" : "string",
           "enum" : [ "AD", "AE", "AF", "AG", "AI", "AL", "AM", "AO", "AP", "AR",
"AT", "AU", "AW", "AZ", "BA", "BB", "BD", "BE", "BF", "BG", "BH", "BI", "BJ",
"BM", "BN", "BO", "BQ", "BR", "BS", "BT", "BV", "BW", "BX", "BY", "BZ", "CA",
"CD", "CF", "CG", "CH", "CI", "CK", "CL", "CM", "CN", "CO", "CR", "CU", "CV",
"CW", "CY", "CZ", "DE", "DJ", "DK", "DM", "DO", "DZ", "EA", "EC", "EE", "EG",
"EH", "EM", "EP", "ER", "ES", "ET", "EU", "FI", "FJ", "FK", "FO", "FR", "GA",
"GB", "GC", "GD", "GE", "GG", "GH", "GI", "GL", "GM", "GN", "GQ", "GR", "GS",
"GT", "GW", "GY", "HK", "HN", "HR", "HT", "HU", "IB", "ID", "IE", "IL", "IM",
"IN", "IQ", "IR", "IS", "IT", "JE", "JM", "JO", "JP", "KE", "KG", "KH", "KI",
"KM", "KN", "KP", "KR", "KW", "KY", "KZ", "LA", "LB", "LC", "LI", "LK", "LR",
"LS", "LT", "LU", "LV", "LY", "MA", "MC", "MD", "ME", "MG", "MK", "ML", "MM",
"MN", "MO", "MP", "MR", "MS", "MT", "MU", "MV", "MW", "MX", "MY", "MZ", "NA",
"NE", "NG", "NI", "NL", "NO", "NP", "NR", "NZ", "OA", "OM", "PA", "PE", "PG",
"PH", "PK", "PL", "PT", "PW", "PY", "QA", "QZ", "RO", "RS", "RU", "RW", "SA",
"SB", "SC", "SD", "SE", "SG", "SH", "SI", "SK", "SL", "SM", "SN", "SO", "SR",
"SS", "ST", "SV", "SX", "SY", "SZ", "TC", "TD", "TG", "TH", "TJ", "TL", "TM",
"TN", "TO", "TR", "TT", "TV", "TW", "TZ", "UA", "UG", "US", "UY", "UZ", "VA",
"VC", "VE", "VG", "VN", "VU", "WO", "WS", "XN", "XU", "XV", "XX", "YE", "ZA",
"ZM", "ZW" ],
           "description" : "Description: This code list is inline with WIPO
Standard ST.3 (two-letter codes for the representation of states, other entities
and organizations) published on September, 2019.; Version: V5_0; AD: Andorra; AE:
United Arab Emirates; AF: Afghanistan; AG: Antigua And Barbuda; AI: Anguilla; AL:
Albania; AM: Armenia; AO: Angola; AP: African Regional Intellectual Property
Organization (ARIPO); AR: Argentina; AT: Austria; AU: Australia; AW: Aruba; AZ:
Azerbaijan; BA: Bosnia and Herzegovina; BB: Barbados; BD: Bangladesh; BE:
Belgium; BF: Burkina Faso; BG: Bulgaria; BH: Bahrain; BI: Burundi; BJ: Benin; BM:
Bermuda; BN: Brunei Darussalam; BO: Bolivia (Plurinational State of); BQ:
Bonaire, Sint Eustatius and Saba; BR: Brazil; BS: Bahamas; BT: Bhutan; BV: Bouvet
Island; BW: Botswana; BX: Benelux Office for Intellectual Property (BOIP); BY:
Belarus; BZ: Belize; CA: Canada; CD: Democratic Republic of the Congo; CF:
Central African Republic; CG: Congo; CH: Switzerland; CI: Côte D'Ivoire; CK: Cook
Islands; CL: Chile; CM: Cameroon; CN: China; CO: Colombia; CR: Costa Rica; CU:
Cuba; CV: Cabo Verde; CW: Curaçao; CY: Cyprus; CZ: Czech Republic; DE: Germany;
DJ: Djibouti; DK: Denmark; DM: Dominica; DO: Dominican Republic; DZ: Algeria; EA:
Eurasian Patent Organization (EAPO); EC: Ecuador; EE: Estonia; EG: Egypt; EH:
Western Sahara; EM: European Union Intellectual Property Office (EUIPO); EP:
European Patent Office (EPO); ER: Eritrea; ES: Spain; ET: Ethiopia; EU: European
Union; FI: Finland; FJ: Fiji; FK: Falkland Islands (Malvinas); FO: Faroe Islands;
FR: France; GA: Gabon; GB: United Kingdom; GC: Patent Office of the Cooperation
Council for the Arab States of the Gulf (GCC Patent Office); GD: Grenada; GE:
Georgia; GG: Guernsey; GH: Ghana; GI: Gibraltar; GL: Greenland; GM: Gambia; GN:
```

```
Guinea; GQ: Equatorial Guinea; GR: Greece; GS: South Georgia and South Sandwich
Islands; GT: Guatemala; GW: Guinea-Bissau; GY: Guyana; HK: Hong Kong, China; HN:
Honduras; HR: Croatia; HT: Haiti; HU: Hungary; IB: International Bureau of the
World Intellectual Property Organization (WIPO); ID: Indonesia; IE: Ireland; IL:
Israel; IM: Isle of Man; IN: India; IQ: Iraq; IR: Iran, Islamic Republic of; IS:
Iceland; IT: Italy; JE: Jersey; JM: Jamaica; JO: Jordan; JP: Japan; KE: Kenya;
KG: Kyrgyzstan; KH: Cambodia; KI: Kiribati; KM: Comoros; KN: Saint Kitts and
Nevis; KP: Democratic People's Republic of Korea; KR: Republic of Korea; KW:
Kuwait; KY: Cayman Islands; KZ: Kazakhstan; LA: Lao People's Democratic Republic;
LB: Lebanon; LC: Saint Lucia; LI: Liechtenstein; LK: Sri Lanka; LR: Liberia; LS:
Lesotho; LT: Lithuania; LU: Luxembourg; LV: Latvia; LY: Libya; MA: Morocco; MC:
Monaco; MD: Republic of Moldova; ME: Montenegro; MG: Madagascar; MK: North
Macedonia; ML: Mali; MM: Myanmar; MN: Mongolia; MO: Macao, China; MP: Northern
Mariana Islands; MR: Mauritania; MS: Montserrat; MT: Malta; MU: Mauritius; MV:
Maldives; MW: Malawi; MX: Mexico; MY: Malaysia; MZ: Mozambique; NA: Namibia; NE:
Niger; NG: Nigeria; NI: Nicaragua; NL: Netherlands; NO: Norway; NP: Nepal; NR:
Nauru; NZ: New Zealand; OA: African Intellectual Property Organization (OAPI);
OM: Oman; PA: Panama; PE: Peru; PG: Papua New Guinea; PH: Philippines; PK:
Pakistan; PL: Poland; PT: Portugal; PW: Palau; PY: Paraguay; QA: Qatar; QZ:
Community Plant Variety Office (European Community) (CPVO); RO: Romania; RS:
Serbia; RU: Russian Federation; RW: Rwanda; SA: Saudi Arabia; SB: Solomon
Islands; SC: Seychelles; SD: Sudan; SE: Sweden; SG: Singapore; SH: Saint Helena,
Ascension and Tristan da Cunha; SI: Slovenia; SK: Slovakia; SL: Sierra Leone; SM:
San Marino; SN: Senegal; SO: Somalia; SR: Suriname; SS: South Sudan; ST: Sao Tome
and Principe; SV: El Salvador; SX: Sint Maarten (Dutch part); SY: Syrian Arab
Republic; SZ: Eswatini; TC: Turks and Caicos Islands; TD: Chad; TG: Togo; TH:
Thailand; TJ: Tajikistan; TL: Timor-Leste; TM: Turkmenistan; TN: Tunisia; TO:
Tonga; TR: Turkey; TT: Trinidad and Tobago; TV: Tuvalu; TW: Taiwan, Province of
China; TZ: United Republic of Tanzania; UA: Ukraine; UG: Uganda; US: United
States of America; UY: Uruguay; UZ: Uzbekistan; VA: Holy See; VC: Saint Vincent
and the Grenadines; VE: Venezuela (Bolivian Republic of); VG: British Virgin
Islands; VN: Viet Nam; VU: Vanuatu; WO: World Intellectual Property Organization
(WIPO) (International Bureau of); WS: Samoa; XN: Nordic Patent Institute (NPI);
XU: International Union for the Protection of New Varieties of Plants (UPOV); XV:
Visegrad Patent Institute (VPI); XX: Unknown states, other entities or
organizations; YE: Yemen; ZA: South Africa; ZM: Zambia; ZW: Zimbabwe"
         }, {
           "type" : "string",
           "enum" : [ "AN", "CS", "DL", "DD", "DT", "RH", "SU", "YD", "YU" ],
           "description" : "Version: V5_0; AN: Netherlands Antilles; CS:
Czechoslovakia; DL: German Democratic Republic; DD: German Democratic Republic;
DT: Federal Republic of Germany; RH: Southern Rhodesia; SU: Soviet Union; YD:
Democratic Yemen; YU: Yugoslavia/ Serbia and Montenegro”
         } ]
        },
        "st13ApplicationNumber" : {
          "type" : "string",
         "pattern" : "\\d{2}\\d{4}\\d{9}",
         "description" : "Description: Application number format recommended in
WIPO Standard ST.13. The sequence of indispensable elements in the application
number format is IP type (2 digits), year designation (4 digits) and serial
number (9 digits).; Version: V5_0”
        },
        "applicationNumberText" : {
          "type" : "string",
         "description" : "Description: Free format of application number;
Version: V5_0"
        }
      },
      "oneOf" : [ {
        "required" : [ "st13ApplicationNumber" ]
      }, {
        "required" : [ "applicationNumberText" ]
      } ]
      }
    }
  }
```

The preferred design approach is to split the data into a set of small components represented by schema modules, which is shown in the new application number schema below. This JSON Schema is built upon smaller JSON Schema modules individually defined in their own schemas.

applicationNumber.json (Example modular schema)

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",
      "description" : "Description: Numbers used by IPOs in order to identify each
application received; Version: V5_0"
    }
  }
}
```

applicationNumberType.json (Example modular schema continued)

```
{
  "$id" : "applicationNumberType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "applicationNumberType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "ipOfficeCode" : {
          "$ref" : "ipOfficeCode.json#/$defs/ipOfficeCode"
        e"st13ApplicationNumber" : {
          "$ref" : "st13ApplicationNumber.json#/$defs/st13ApplicationNumber"
        },
        "applicationNumberText" : {
          "$ref" : "applicationNumberText.json#/$defs/applicationNumberText"
        }
      },
      "oneOf" : [ {
        "required" : [ "st13ApplicationNumber" ]
      }, {
        "required" : [ "applicationNumberText" ]
      } ]
    }
  }
}
```

ipOfficeCode.json (Example modular schema continued)

```
{
  "$id" : "ipOfficeCode.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "ipOfficeCode" : {
      "$ref" : "#/$defs/ipOfficeCode"
    }
  },
  "required" : [ "ipOfficeCode" ],
  "$defs" : {
    "ipOfficeCode" : {
      "$ref" : "extendedWIPOST3CodeType.json#/$defs/extendedWIPOST3CodeType",
      "description" : " Description: Two-letter alphabetic codes which represent
the names of states, other entities and intergovernmental organizations the
legislation of which provides for the protection of IP rights or which
organizations are acting in the framework of a treaty in the field of IP;
Version: V5_0"
    }
  }
}
```

extendedWIPOST3CodeType.json（Example modular schema continued)

```
{
  "$id" : "extendedWIPOST3CodeType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "extendedWIPOST3CodeType" : {
      "description" : "Version: V5_0",
      "anyOf" : [ {
        "$ref" : "wipoST3CodeType.json#/$defs/wipoST3CodeType"
      }, {
        "$ref" : "wipoFormerST3CodeType.json#/$defs/wipoFormerST3CodeType"
      } ]
    }
  }
}
```

wipoST3CodeType.json (Example modular schema continued)

```
{
  "$id" : "wipoST3CodeType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "wipoST3CodeType" : {
      "description" : "Description: This code list is inline with WIPO Standard
ST.3 (two-letter codes for the representation of states, other entities and
organizations) published on September, 2019.; Version: V5_0; AD: Andorra; AE:
United Arab Emirates; AF: Afghanistan; AG: Antigua And Barbuda; AI: Anguilla;
AL: Albania; AM: Armenia; AO: Angola; AP: African Regional Intellectual Property
Organization (ARIPO); AR: Argentina; AT: Austria; AU: Australia; AW: Aruba; AZ:
Azerbaijan; BA: Bosnia and Herzegovina; BB: Barbados; BD: Bangladesh; BE:
Belgium; BF: Burkina Faso; BG: Bulgaria; BH: Bahrain; BI: Burundi; BJ: Benin;
BM: Bermuda; BN: Brunei Darussalam; BO: Bolivia (Plurinational State of); BQ:
Bonaire, Sint Eustatius and Saba; BR: Brazil; BS: Bahamas; BT: Bhutan; BV:
Bouvet Island; BW: Botswana; BX: Benelux Office for Intellectual Property
(BOIP); BY: Belarus; BZ: Belize; CA: Canada; CD: Democratic Republic of the
Congo; CF: Central African Republic; CG: Congo; CH: Switzerland; CI: Côte
D'Ivoire; CK: Cook Islands; CL: Chile; CM: Cameroon; CN: China; CO: Colombia;
CR: Costa Rica; CU: Cuba; CV: Cabo Verde; CW: Curaçao; CY: Cyprus; CZ: Czech
Republic; DE: Germany; DJ: Djibouti; DK: Denmark; DM: Dominica; DO: Dominican
Republic; DZ: Algeria; EA: Eurasian Patent Organization (EAPO); EC: Ecuador; EE:
Estonia; EG: Egypt; EH: Western Sahara; EM: European Union Intellectual Property
Office (EUIPO); EP: European Patent Office (EPO); ER: Eritrea; ES: Spain; ET:
Ethiopia; EU: European Union; FI: Finland; FJ: Fiji; FK: Falkland Islands
(Malvinas); FO: Faroe Islands; FR: France; GA: Gabon; GB: United Kingdom; GC:
Patent Office of the Cooperation Council for the Arab States of the Gulf (GCC
Patent Office); GD: Grenada; GE: Georgia; GG: Guernsey; GH: Ghana; GI:
Gibraltar; GL: Greenland; GM: Gambia; GN: Guinea; GQ: Equatorial Guinea; GR:
Greece; GS: South Georgia and South Sandwich Islands; GT: Guatemala; GW: Guinea-
Bissau; GY: Guyana; HK: Hong Kong, China; HN: Honduras; HR: Croatia; HT: Haiti;
HU: Hungary; IB: International Bureau of the World Intellectual Property
Organization (WIPO); ID: Indonesia; IE: Ireland; IL: Israel; IM: Isle of Man;
IN: India; IQ: Iraq; IR: Iran, Islamic Republic of; IS: Iceland; IT: Italy; JE:
Jersey; JM: Jamaica; JO: Jordan; JP: Japan; KE: Kenya; KG: Kyrgyzstan; KH:
Cambodia; KI: Kiribati; KM: Comoros; KN: Saint Kitts and Nevis; KP: Democratic
People's Republic of Korea; KR: Republic of Korea; KW: Kuwait; KY: Cayman
Islands; KZ: Kazakhstan; LA: Lao People's Democratic Republic; LB: Lebanon; LC:
Saint Lucia; LI: Liechtenstein; LK: Sri Lanka; LR: Liberia; LS: Lesotho; LT:
Lithuania; LU: Luxembourg; LV: Latvia; LY: Libya; MA: Morocco; MC: Monaco; MD:
Republic of Moldova; ME: Montenegro; MG: Madagascar; MK: North Macedonia; ML:
Mali; MM: Myanmar; MN: Mongolia; MO: Macao, China; MP: Northern Mariana Islands;
MR: Mauritania; MS: Montserrat; MT: Malta; MU: Mauritius; MV: Maldives; MW:
Malawi; MX: Mexico; MY: Malaysia; MZ: Mozambique; NA: Namibia; NE: Niger; NG:
Nigeria; NI: Nicaragua; NL: Netherlands; NO: Norway; NP: Nepal; NR: Nauru; NZ:
New Zealand; OA: African Intellectual Property Organization (OAPI); OM: Oman;
PA: Panama; PE: Peru; PG: Papua New Guinea; PH: Philippines; PK: Pakistan; PL:
Poland; PT: Portugal; PW: Palau; PY: Paraguay; QA: Qatar; QZ: Community Plant
Variety Office (European Community) (CPVO); RO: Romania; RS: Serbia; RU: Russian
Federation; RW: Rwanda; SA: Saudi Arabia; SB: Solomon Islands; SC: Seychelles;
SD: Sudan; SE: Sweden; SG: Singapore; SH: Saint Helena, Ascension and Tristan da
Cunha; SI: Slovenia; SK: Slovakia; SL: Sierra Leone; SM: San Marino; SN:
Senegal; SO: Somalia; SR: Suriname; SS: South Sudan; ST: Sao Tome and Principe;
SV: El Salvador; SX: Sint Maarten (Dutch part); SY: Syrian Arab Republic; SZ:
Eswatini; TC: Turks and Caicos Islands; TD: Chad; TG: Togo; TH: Thailand; TJ:
Tajikistan; TL: Timor-Leste; TM: Turkmenistan; TN: Tunisia; TO: Tonga; TR:
Turkey; TT: Trinidad and Tobago; TV: Tuvalu; TW: Taiwan, Province of China; TZ:
United Republic of Tanzania; UA: Ukraine; UG: Uganda; US: United States of
America; UY: Uruguay; UZ: Uzbekistan; VA: Holy See; VC: Saint Vincent and the
Grenadines; VE: Venezuela (Bolivian Republic of); VG: British Virgin Islands;
VN: Viet Nam; VU: Vanuatu; WO: World Intellectual Property Organization (WIPO)
(International Bureau of); WS: Samoa; XN: Nordic Patent Institute (NPI); XU:
International Union for the Protection of New Varieties of Plants (UPOV); XV:
```

```
Visegrad Patent Institute (VPI); XX: Unknown states, other entities or
organizations; YE: Yemen; ZA: South Africa; ZM: Zambia; ZW: Zimbabwe",
      "type" : "string",
      "enum" : [ "AD", "AE", "AF", "AG", "AI", "AL", "AM", "AO", "AP", "AR",
"AT", "AU", "AW", "AZ", "BA", "BB", "BD", "BE", "BF", "BG", "BH", "BI", "BJ",
"BM", "BN", "BO", "BQ", "BR", "BS", "BT", "BV", "BW", "BX", "BY", "BZ", "CA",
"CD", "CF", "CG", "CH", "CI", "CK", "CL", "CM", "CN", "CO", "CR", "CU", "CV",
"CW", "CY", "CZ", "DE", "DJ", "DK", "DM", "DO", "DZ", "EA", "EC", "EE", "EG",
"EH", "EM", "EP", "ER", "ES", "ET", "EU", "FI", "FJ", "FK", "FO", "FR", "GA",
"GB", "GC", "GD", "GE", "GG", "GH", "GI", "GL", "GM", "GN", "GQ", "GR", "GS",
"GT", "GW", "GY", "HK", "HN", "HR", "HT", "HU", "IB", "ID", "IE", "IL", "IM",
"IN", "IQ", "IR", "IS", "IT", "JE", "JM", "JO", "JP", "KE", "KG", "KH", "KI",
"KM", "KN", "KP", "KR", "KW", "KY", "KZ", "LA", "LB", "LC", "LI", "LK", "LR",
"LS", "LT", "LU", "LV", "LY", "MA", "MC", "MD", "ME", "MG", "MK", "ML", "MM",
"MN", "MO", "MP", "MR", "MS", "MT", "MU", "MV", "MW", "MX", "MY", "MZ", "NA",
"NE", "NG", "NI", "NL", "NO", "NP", "NR", "NZ", "OA", "OM", "PA", "PE", "PG",
"PH", "PK", "PL", "PT", "PW", "PY", "QA", "QZ", "RO", "RS", "RU", "RW", "SA",
"SB", "SC", "SD", "SE", "SG", "SH", "SI", "SK", "SL", "SM", "SN", "SO", "SR",
"SS", "ST", "SV", "SX", "SY", "SZ", "TC", "TD", "TG", "TH", "TJ", "TL", "TM",
"TN", "TO", "TR", "TT", "TV", "TW", "TZ", "UA", "UG", "US", "UY", "UZ", "VA",
"VC", "VE", "VG", "VN", "VU", "WO", "WS", "XN", "XU", "XV", "XX", "YE", "ZA",
"ZM", "ZW" ]
    }
  }
}
```

wipoFormerST3CodeType.json (Example modular schema continued)

```
{
  "$id" : "wipoFormerST3CodeType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "wipoFormerST3CodeType" : {
      "description" : "Version: V5_0; AN: Netherlands Antilles; CS:
Czechoslovakia; DL: German Democratic Republic; DD: German Democratic Republic;
DT: Federal Republic of Germany; RH: Southern Rhodesia; SU: Soviet Union; YD:
Democratic Yemen; YU: Yugoslavia/ Serbia and Montenegro",
      "type" : "string",
      "enum" : [ "AN", "CS", "DL", "DD", "DT", "RH", "SU", "YD", "YU" ]
    }
  }
}
```

st13ApplicationNumber.json (Example modular schema continued)

```
{
  "$id" : "st13ApplicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "st13ApplicationNumber" : {
      "$ref" : "#/$defs/st13ApplicationNumber"
    }
  },
  "required" : [ "st13ApplicationNumber" ],
  "$defs" : {
    "st13ApplicationNumber" : {
      "$ref" :
"st13ApplicationNumberType.json#/$defs/st13ApplicationNumberType",
      "description" : "Description: Application number format recommended in
WIPO Standard ST.13. The sequence of indispensable elements in the application
number format is IP type (2 digits), year designation (4 digits) and serial
number (9 digits).; Version: V5_0"
    }
  }
}
```

st13ApplicationNumberType.json (Example modular schema continued)

```
{
  "$id" : "st13ApplicationNumberType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "st13ApplicationNumberType" : {
      "description" : "Version: V5_0",
      "type" : "string",
      "pattern" : "\\d{2}\\d{4}\\d{9}"
    }
  }
}
```

applicationNumberText.json (Example modular schema continued)

```
{
  "$id" : "applicationNumberText.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumberText" : {
      "$ref" : "#/$defs/applicationNumberText"
    }
  },
  "required" : [ "applicationNumberText" ],
  "$defs" : {
    "applicationNumberText" : {
      "type" : "string",
      "description" : "Description: Free format of application number; Version:
V5_0"
    }
  }
}
```

JSON Schemas should use the "`$defs`" keyword to create global definitions for properties and their contents that can be reused, as shown in the above example. This is roughly equivalent to creating global element declarations and named types in XML schema.

[JSD-04] JSON Schemas SHOULD use the "`$defs`" keyword that includes a reusable definition for each property and property type.

[JSD-05] Developers MUST use existing JSON Schemas that are defined in Annex II to this draft standard, wherever applicable, prior to creating new JSON Schemas.

[JSD-06] Developers SHOULD create new JSON Schemas only after determining that no existing JSON Schemas adequately describe the given construct.

6.3 Documentation

JSON Schemas should be self-descriptive. Developers should aim to make JSON construct names meaningful. In addition, the JSON Schema should have documentation describing the schema and the JSON constructs.

To promote reusability by keeping it general, the JSON Schema should not provide documentation on system specific implementation details.

[JSD-07] Documentation SHOULD NOT describe implementation details or other information not directly related to the meaning of the construct.

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

[JSD-08] JSON Schemas SHOULD include JSON Schema header documentation using the "`description`" keyword.

[JSD-09] The items listed in Table 1 below SHOULD be included in the header section of all JSON schemas.

Table 1. JSON Schema header documentation items

Header item name	Description	Required/Optional
`Description`	Plain text description of the information described by the schema	Required except JSON Schemas originated from simple Type of ST.96 XSDs for which a description is not provided such as `DateType`.
`Version`	Major and Minor version number of the schema	Required
`SchemaCreatedDate`	Date of schema created	Optional
`SchemaLastModifiedDate`	Date of schema last modified	Optional
`SchemaContactPoint`	Name of Organization to contact with questions about the schema	Optional
`SchemaReleaseNoteURL`	Location where schema release notes are published	Optional

[JSD-10] The header documentation items such as Published on and version number above SHOULD be separated by semicolons, with spaces allowed after the semicolon, and make up the value associated with the "`description`" keyword. If a value is not available for the header item then just the label should be included, as shown in the following example:

An example of header documentation for applicationBody (Document Level Schema)
`"description" : "Description: Body of a patent application; Version: V5_0; SchemaCreatedDate: 2012-07-13; SchemaLastModifiedDate: 2021-10-01; SchemaContactPoint: xml.standards@wipo.int; SchemaReleaseNoteURL: http://www.wipo.int/standards/XMLSchema/ST96/V5_0/ReleaseNotes.pdf"`

An example of header documentation for ipOfficeCode (Non Document Level Schema)
`"description" : " Description: Two-letter alphabetic codes which represent the names of states, other entities and intergovernmental organizations the legislation of which provides for the protection of IP rights or which organizations are acting in the framework of a treaty in the field of IP; Version: V5_0"`

An example of header documentation for appellateBodyCategoryType.json (Enumeration Type definitions Schema)
`"description" : "Version: V5_0; Office appeal board: Appeal board within the IP office; Court: Court; Appeal Court: Second instance court; Supreme Court: Highest appellate court"`

An example of header documentation for Type definitions Schema
`"description" : "Version: V5_0"`

6.4 Filename

The rules from ST.96 are followed for the JSON Schema filename with the exception that they should be LCC.

Schema filenames and schema names are often paired. Schema filenames rely on the corresponding schema names. For example, the filename of `postalAddressType.json` is derived from the schema name `postalAddressType`. Thus, schema file naming conventions are related to the rules for JSON naming conventions in this Standard.

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.

[JSD-11] The characters used in Schema filenames MUST belong to the following set: ‘a-z, A-Z, 0-9, underscore “_”, and period “.”.

[JSD-12] 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.json`, `languageCode.json, applicationBody_V1_0.json`.

[JSD-13] 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, , `trademarkApplication_V1_1_D1.json`. 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 IP domain and a minor version number in the draft schema file SHOULD be zero “0”.

6.5 JSON Schema Properties Structuring

JSON Schemas should have property `"type": "object"` to ensure that JSON is used only for nested structures, not individual values. Taken from `applicationNumber.json` example below:

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",
      "description" : "Description: Numbers used by IPOs in order to identify each 
application received; Version: V5_0"
    }
  }
}
```

[JSD-14] The outermost schema object MUST have a "`type`" keyword whose value is "`object`".

[JSD-15] The outermost schema object MUST have a "`$defs`" keyword whose value is the property of the outermost schema object.

[JSD-16] The outermost schema object MUST have a "`required`" keyword whose value is an array that contains a single item, i.e., the property of the outermost schema object.

JSON Schema extensions (customizations) may be used. If the extended type is an object type then it must be referenced instead of duplicating its properties to promote reusability.

[JSD-17] JSON Schema extensions (customizations) for object types MUST be implemented by referencing the JSON Schema of the extended type.

7. JSON SCHEMA CONSTRUCTS DESIGN RULES

7.1 Overview

This section establishes the rules for JSON Schema constructs, specifically arrays, objects and primitive values. Standardization of names for schema constructs are critical to the development of a robust data architecture.

7.2 Properties

Properties, also known as members, are the basic building blocks of a JSON construct.

[JSC-01] Definitions SHOULD use existing schemas to the maximum extent possible.

[JSC-02] Multiple properties that can be logically grouped together MAY be declared in a single schema file under the global definition.

7.3 Definitions

Each property should have a global definition that is defined in its JSON Schema. This will allow the property name to be reused in many parents and have a consistent definition across all of them. Please see the "`applicationNumber`" property in the example below.

[JSC-03] Each property listed in a properties keyword SHOULD refer to a global definition that is defined within the "`$defs`" keyword. That global definition SHOULD have the same name as the property.

An example of a property referring to a global definition

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",
      "description" : "Description: Numbers used by IPOs in order to identify each 
application received; Version: V5_0"
    }
  }
}
```

The global definition of a property should consist of the file name and description of the property.

[JSC-04] The global definition of a property SHOULD consist of the filename and "`description`" of the property.

Properties must have types. They can either be defined directly if they are primitive types (except object), or be handled through a reference to a global property definition in another JSON Schema.

[JSC-05] A property MUST have a type that is specified using the "`type`" keyword, either as a direct property or through a reference to a global definition.

7.4 Type Definitions

JSON Schemas may define reusable type definitions that are referenced from global property definitions. These global type definitions should consist of a "`type`" keyword, "`properties`" keyword (if type is "`object`") and any other value constraints.

[JSC-06] A schema MAY define global type definitions in order to reuse content models across many properties.

A reusable type definition

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",
      "description" : "Description: Numbers used by IPOs in order to identify each
application received; Version: 5_0"
    }
  }
}
```

[JSC-07] Definitions that represent types MUST have names that are in LCC convention + Suffix "Type".

7.5 JSON Primitive Type

[JSC-08] The most specific JSON primitive type that is relevant SHOULD be used for a property. Primitive types include: “`string`”, “`number`”, “`integer`”, “`object`”, “`array`”, “`boolean`”, and “`null`”.

For example, if a property value will be an integer, the type "`integer`" should be used rather than the more generic "`number`" or the more permissive "`string`". For “`string`” type, the built-in formats should be used if applicable, e.g. “`date-time`” or “`duration`”.

7.6 Code Lists

In certain cases, it is advantageous to restrict a value to an enumerated list of codes that are standard and acceptable for data exchange purposes. Code lists are a means to create a controlled vocabulary of permitted values for a data element (e.g., a standard code list for country codes, language codes, IP Office codes, etc.). Code lists which already exist in the public domain and are maintained by relevant standards committee such as ISO should be used.

[JSC-09] WIPO Standard ST.3 MUST be used for representing IPOs, states, other entities, organizations and for priority and designated country/organization.

[JSC-10] ISO 3166-1-Alpha-2 Code Elements (two-letter country codes) MUST be used for the representation of the names of countries for addressing and citizenship.

[JSC-11] ISO 639-1 (two-Letter Language Codes) MUST be used for Language Codes.

[JSC-12] ISO 4217-Alpha (three-Letter Currency Codes) MUST be used for Currency Codes.

[JSC-13] The JSON `enum` keyword SHOULD be used for defining the code lists.

[JSC-14] The characters used in enumeration values MUST be restricted to the following set: `{a-z, A-Z, 0-9`, period (.), comma (,), spaces, dash (-) and underscore (_)}

7.7 Arrays

The term cardinality is defined as the number of items in an array. Cardinality is indicated in a schema using the `minItems` and `maxItems` keywords. It is recommended that schema developers not specify default values for occurrence indicators (i.e., "`minItems`": 0) because doing so can unnecessarily clutter a schema.

[JSC-15] JSON Schemas SHOULD use `minItems` and `maxItems` keywords for arrays, except for the default value of `minItems` (0).

The type of items in an array must be defined using the "`items`" keyword. For simplicity, all items in an array must have the same type. If a sequence of objects of different types is desired, they should be defined as separate properties of an object.

[JSC-16] For each object of type array, there MUST be an "`items`" keyword and its value MUST be a single schema object and not an array. All items in an array MUST have the same type.

The "`additionalItems`" keyword must not be used for arrays, since it is not relevant when the value of "`items`" is a single schema object.

[JSC-17] The "`additionalItems`" keyword SHOULD NOT be used when “`items`” is a single schema object.

7.8 Objects

7.8.1 Property "wildcards"

JSON Schemas should not allow arbitrary properties to be part of the JSON instance and still be valid, which can corrupt the integrity of the data exchange.

Use of the "`additionalProperties`" keyword is required and it must be set to "`false`". Otherwise, undefined properties will be permitted in instances.

[JSC-18] A JSON Schema MUST use "`additionalProperties`" and set its value to "`false`" for every object.

Using the "`patternProperties`" keyword is not allowed. This keyword allows mapping of regular expressions to schemas. For example, it allows implicit schema definitions based on the name of the property. Given a particular kind of property name, a particular schema is applied.

[JSC-19] A schema MUST NOT use the "`patternProperties`" keyword.

7.8.2 Order of Properties

JSON Schema does not enforce a particular order on properties of an object. However, if a JSON Schema has a corresponding XML schema, it is recommended that the properties be listed in the same order as in the XML schema in both the JSON Schema and the instance.

[JSC-20] A schema SHOULD use the same order of properties as in the corresponding XML schema, if one exists.

8 JSON SCHEMA IDENTIFIERS

8.1 Overview

An ID in a JSON Schema provides a URI that identifies a category of information based on the business domain (e.g., enterprise, patents, and trademarks). This draft standard has opted to use many-to-one relationships between the handfuls of IDs and perhaps hundreds of JSON constructs. A group of related JSON constructs with unique names are going to be associated with a particular ID. A uniform resource identifier (URI) should be used for identification.

[JID-01] IDs MUST be used in schemas using the "`$id`" keyword.

9. JSON INSTANCE DESIGN RULES

The JSON Schema defines the structure and constraints for the JSON instance. To enhance and ensure sound (intra and inter office) data exchange, JSON instances should be associated with the JSON Schema to ensure validity and conformance.

9.1 Order of Properties

JSON Schema does not enforce a particular order on properties of an object. However, it is recommended that the properties appear in the instance in the same order as in the JSON Schema.

[JIN-01] A JSON instance document SHOULD use the same order of properties as in the corresponding JSON Schema if one exists.

9.2 JSON instance validation

Successful validation of JSON instances ensures that its content satisfies all the requirements defined in the corresponding schemas.

[JIN-02] JSON instance documents MAY be validated against a corresponding schema during processing.

[JIN-03] A run-time schema MAY be created to meet performance requirements of the application in the run-time environment. For example, the compound schema for application number may be used as a run-time schema.

[Note: The run-time JSON schema is similar to ST.96 flattened schema which is actually used in production environment for performance and other concerns. So we think that the run-time JSON schema would serve the similar purpose. However, at the time of preparation of this Standard there are no such practices or support from JSON industry available tools/editors.]

[JIN-04] All modifications, updates, revisions, and new releases MUST first be submitted to the XML4IP Task Force for approval before the changes can be incorporated into the run-time schema.

It is desirable for a JSON instance to conform to a schema.

[JIN-05] A JSON instance SHOULD conform to a particular JSON Schema that conforms to the rules described in this Standard.

8. REFERENCES

WIPO Standards

– WIPO Standard ST.96:	Processing of Intellectual Property information using XML
– WIPO Standard ST.90:	Recommendation for Processing and communicating Intellectual Property Data using Web APIs

Standards and Industry Specifications

– JSON Specification:	http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
– JSON Schema Core, 2020-12 draft:	http://json-schema.org/latest/json-schema-core.html
– JSON Schema Validation, 2020-12 draft:	http://json-schema.org/latest/json-schema-validation.html
– OpenAPI v3.1.0 Specification:	https://spec.openapis.org/oas/v3.1.0#schema
– ISO 21778 Standard ECMA-404 – The JSON Data Interchange Syntax:	https://www.ecma-international.org/publications/standards/Ecma-404.htm
– UBL-2.1 Universal Business Language:	http://docs.oasis-open.org/ubl/UBL-2.1.html
– ISO 3166 Country Codes:	https://www.iso.org/iso-3166-country-codes.html
– ISO 639 Language Codes:	https://www.iso.org/iso-639-language-codes.html
– ISO 4217 Current Codes:	https://www.iso.org/iso-4217-currency-codes.html
– ISO 11179:	https://www.iso.org/standard/60341.html
– RFC 2119:	https://www.ietf.org/rfc/rfc2119.txt

[Annex I follows]

ST.XX - ANNEX I

TRANSFORMATION RULES FROM ST.96 XSD TO JSON SCHEMA AND GUIDELINES FOR USE

INTRODUCTION

JSON has been gradually adopted and used by Intellectual Property Offices (IPOs) and IP industry while XML (eXtensible Markup Language) based on WIPO XML Standards is still widely used. WIPO Standard ST.96 recommends the XML resources to be used for filing, publication, processing, and exchange of information for various ~~all~~ types of intellectual property (IP), i.e., patents, trademarks, industrial designs, geographical indications and copyright. Standard ST.96 is implemented by IPOs as published or customized as required.

For the facilitation of data exchange between IPOs and data dissemination by IPOs in two formats, i.e., XML and JSON, data consistency ~~and~~, data structure and tag name compatibility between the two formats is important. This data consistency and compatibility can be achieved using the compatible XML Schemas and JSON Schemas which will be used to validate XML instances and JSON instances respectively.

Annex I is intended to provide rules and guidelines for transformations of ST.96 XSDs to corresponding JSON Schemas. JSON Schemas provided in Annex II of ST.XX are the result of the application of these transformation rules and it is recommended that IPOs follow these when they develop JSON Schemas, which will be consistent and compatible with their customized ~~tailored~~ ST.96 XSDs.

Annex I also includes a transformation tool, as an Appendix, which applies these rules in for use by those IPOs which wish to develop their own JSON Schemas based on their customized ST.96 XSDs.

At the time of preparation, the 2020-12 is the latest draft version of the JSON Schema and the version 5.0 is the latest ST.96. Therefore, the Transformation Rules and Guidelines are based on the JSON 2022-12 version and ST.96 version 5.0 which uses XSD 1.1. The rules provided should be aligned with the rules defined in the Main Body of the Standard ST.XX. Transformation rules are identified in this Annex using the prefix 'TR'.

TRANSFORMATION ALGORITHM AND GUIDELINES

In order to transform ST.96 XSDs to JSON Schemas, the following algorithm should apply:

1. Transform the XSD filename to JSON Schema filename; see Filename Rules in Section 6.4
2. Transform a XSD fragment to a compatible JSON Schema fragment ~~by using either:~~

    ~~(a) absolute XPaths; or~~

    ~~(b) relative Xpaths.~~

It should be noted that the transformation of the XSDs to JSON Schemas does not require any strict order. That said, the validation of the generated JSON Schemas should be triggered after all of them are generated because a generated JSON Schema file may have dependencies "`$ref`" that are still not generated.

SCOPE

As the JSON S~~s~~chemas published as part of this Standard result from the transformation of ST.96 XSDs, this Annex I covers only XSD constructors, keywords and others used in WIPO Standard ST.96.

FILENAME TRANSFORMATION

[TR-01] JSON Schema filenames and JSON Schema object or property names MUST be the same with the corresponding ST.96 XSD filenames or component names but following Rules defined in Section 5.2 "JSON Naming Conventions of this Standard" and using the suffix “.json” instead of “.xsd” in the case of filenames.

NAMESPACES TRANSFORMATION

XSD namespaces should not be preserved in JSON Schemas. It should be noted that the native namespace feature is not supported in 2022-12 JSON Schema. If this feature is supported by a JSON Schema standard in the future, the XSD namespace transformation will be reconsidered. However, the namespace emulation can only be done if JSON schemas are stored in sub-folders per IP domain. The design of ST.96 provides each IP domain in separate sub-folders.

[TR-02] For the XSDs’ namespace emulation, JSON schemas MUST be stored in sub-folders per IP domain and Common, i.e., Common, Copyright, Design, GeographicalIndication, Patent, Trademark, and ExternalStandards as the corresponding XSDs are stored in ST.96.

BUILT-IN DATA TYPES TRANSFORMATION

According to JSC-09, the most specific JSON primitive type that is relevant SHOULD be used for a property. For example, if a property value will be an integer, the type "`integer`" should be used rather than the more generic "`number`" or the more permissive "`string`".

Table 2 provides one-to-one mapping of a XSD data type to a JSON data type (built-in or custom / inline or in a separate file) which are used for transformation of ST.96 XSD to JSON Schema.

[TR-03] The XSD Data Types listed in the following table SHOULD be transformed to the corresponding JSON Schema Data Types as defined in Table 2.

Table 2: Transformation of simple XSD data types

XSD Data Type	JSON Schema Data Type or JSON Schema Type Definition
`xsd:string`	`"type": "string"`
`Token`	`"type": "string"`
`xsd:integer`	`"type": "integer"`

`xsd:float`
`xsd:double`
`xsd:decimal`
	`"type": "number"`

`xsd:Boolean`	`"type": "boolean"`

`xsd:positiveInteger`	`"type": "integer",`
	`"minimum": 0,`
	`"exclusiveMinimum": true`

`xsd:negativeInteger`	`"type": "integer",`
	`"maximum": 0,`
	`"exclusiveMaximum": true`

`xsd:nonPositiveInteger`	`"type": "integer",`
	`"maximum": 0,`
	`"exclusiveMaximum": false`

`xsd:nonNegativeInteger`	`"type": "integer",`
	`"minimum": 0,`
	`"exclusiveMinimum": false`

`xsd:date,`
`xsd:dateTime,`
`xsd:time`
	`"type": "string",`
	`"format": "date-time"`

`gYearMonth`	No built-in type exists. So a specific file “gYearMonth.json” is defined.
	`"gYearMonth": {`
	`  "anyOf": [`
	`    {`
	`      "type": "object",`
	`      "properties": {`
	`        "year": {`
	`          "type": "integer"`
	`        },`
	`        "month": {`
	`          "type": "integer",`
	`          "minimum": 1,`
	`          "maximum": 12`
	`        },`
	`        "timezone": {`
	`          "type": "integer",`
	`          "minimum": -1440,`
	`          "maximum": 1439`
	`        }`
	`      }`
	`    }`
	`  ]`
	`}`

`gYear`	No built-in type exists. So a specific file “gYear.json” is defined.
	`"gYear": {`
	`      "anyOf": [`
	`        {`
	`          "type": "object",`
	`          "properties": {`
	`            "year": {`

	`"type": "integer"`
	`},`
	`"timezone": {`
	`"type": "integer",`
	`"minimum": -1440,`
	`"maximum": 1439`
	`}`
	`}`
	`}`
	`]`
	`}`

`xsd:anyURI`	`{`
	`"type": "string",`
	`"format": "uri"`
	`}`

XSD DEFINITION

To transform XSD definition, the "`$schema`" object property wrapper to current component definition should be added. The attributes of Root element of a XML Schema "`xsd:schema`" do not have equivalent placeholders in JSON Schema.

[TR-04] `xsd:schema` and its attribute values SHOULD be transformed to appropriate JSON properties from source XSDs as guided below.

XSD	JSON	Notes
`xsd:schema`	`"$schema"`	
`@xmlns`	~~ignored~~N/A	ignored
`@targetNamespace`	~~ignored~~N/A	ignored
`@elementFormDefault`	~~ignored~~N/A	ignored
`@attributeFormDefault`	~~ignored~~N/A	ignored
`@version`	`"description"`	Concatenated with description value with "`Version:` " label. Please refer to JSD-10 and TR-20.

Some examples are provided below:

Component XML Schema definition (AbstractNumber.xsd)

```
<?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="V5_0">
<xsd:element name="AbstractNumber" type="xsd:string">
       <xsd:annotation>
           <xsd:documentation>Number assigned to an abstract published without the
full document in a collection of abstracts. This collection can be a journal,
conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
Abstracts), etc.</xsd:documentation>
       </xsd:annotation>
</xsd:element>
</xsd:schema>
```

Component JSON Schema definition (abstractNumber.json)

```
{
  "$id" : "abstractNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "additionalProperties" : false,
  "properties" : {
    "abstractNumber" : {
      "$ref" : "#/$defs/abstractNumber"
    }
  },
  "required" : [ "abstractNumber" ],
  "$defs" : {
    "abstractNumber" : {
      "type" : "string",
      "description" : "Description: Number assigned to an abstract published without
the full document in a collection of abstracts. This collection can be a journal,
conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
Abstracts), etc.; Version: V5_0"
    }
  }
}
```

Component Type XML Schema definition (AdditionalRemarkType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="languageCode.xsd"/>
   <xsd:complexType name="AdditionalRemarkType">
      <xsd:sequence>
         <xsd:element ref="com:P"/>
      </xsd:sequence>
      <xsd:attribute ref="com:languageCode"/>
   </xsd:complexType>
</xsd:schema>
```

Component Type JSON Schema definition (additionalRemarkType.json)

```
{
  "$id" : "additionalRemarkType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "additionalRemarkType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "languageCode" : {
          "$ref" : "languageCode.json#/$defs/languageCode"
        },
        "p" : {
          "$ref" : "p.json#/$defs/p"
        }
      },
      "required" : [ "p" ]
    }
  }
}
```

Document level Component XML Schema definition (DesignApplication_V5_0.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:dgn="http://www.wipo.int/standards/XMLSchema/ST96/Design"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
   <xsd:annotation>
      <xsd:appinfo>
         <com:SchemaCreatedDate>2012-07-13</com:SchemaCreatedDate>
         <com:SchemaLastModifiedDate>2021-10-01</com:SchemaLastModifiedDate>
         <com:SchemaContactPoint>xml.standards@wipo.int</com:SchemaContactPoint>

   <com:SchemaReleaseNoteURL>http://www.wipo.int/standards/XMLSchema/ST96/V5_0/Release
Notes.pdf</com:SchemaReleaseNoteURL>
```

```
        </xsd:appinfo>
    </xsd:annotation>
<xsd:include schemaLocation="DesignApplicationType_V5_0.xsd"/>
    <xsd:element name="DesignApplication" type="dgn:DesignApplicationType">
        <xsd:annotation>
            <xsd:documentation>Details on a design application</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

Document level Component JSON Schema definition (designApplication_V5_0.json)

```
{
  "$id" : "designApplication_V5_0.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "designApplication" : {
      "$ref" : "#/$defs/designApplication"
    }
  },
  "required" : [ "designApplication" ],
  "$defs" : {
    "designApplication" : {
      "$ref" : "designApplicationType_V5_0.json#/$defs/designApplicationType",
      "description" : "Description: Details on a design application; Version: V5_0;
SchemaCreatedDate: 2012-07-13; SchemaLastModifiedDate: 2021-10-01;
SchemaContactPoint: xml.standards@wipo.int; SchemaReleaseNoteURL:
http://www.wipo.int/standards/XMLSchema/ST96/V5_0/ReleaseNotes.pdf"
    }
  }
}
```

Document level Component Type XML Schema definition (DesignApplicationType_V5_0.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:dgn="http://www.wipo.int/standards/XMLSchema/ST96/Design"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RequestSoftware.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationFormName.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RequestExamination.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RegistrationOfficeCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReceivingOfficeCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReceivingOfficeDate.xsd"/>
    <xsd:include schemaLocation="../ReceiptNumber.xsd"/>
    <xsd:include schemaLocation="../SealedDepositIndicator.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationNumber.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/FilingPlace.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicantFileReference.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationLanguageCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/SecondLanguageCode.xsd"/>
    <xsd:include schemaLocation="../DesignTotalQuantity.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/CorrespondenceLanguageCode.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationCurrentStatusCategory.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationCurrentStatusDate.xsd"/>
    <xsd:include schemaLocation="../DesignatedCountryBag.xsd"/>
    <xsd:include schemaLocation="../DesignBag.xsd"/>
    <xsd:include schemaLocation="../ApplicantBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RepresentativeBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/Authorization.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationEventBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/DocumentIncludedBag.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationStatementBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/PaymentBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReimbursementBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/SignatureBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationDate.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationDateTime.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/CorrespondenceAddress.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/BusinessEntityStatusCategory.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/InternationalRegistrationNumber.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/operationCategory.xsd"/>
    <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="DesignApplicationType">
        <xsd:sequence>
            <xsd:element ref="com:RequestSoftware" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationFormName" minOccurs="0"/>
            <xsd:element ref="com:RequestExamination" minOccurs="0"/>
            <xsd:element ref="com:RegistrationOfficeCode"/>
            <xsd:element ref="com:ReceivingOfficeCode" minOccurs="0"/>
            <xsd:element ref="com:ReceivingOfficeDate" minOccurs="0"/>
            <xsd:element ref="dgn:ReceiptNumber" minOccurs="0"/>
            <xsd:element ref="dgn:SealedDepositIndicator" minOccurs="0"/>
            <xsd:element ref="com:ApplicationNumber" minOccurs="0"/>
            <xsd:element ref="com:InternationalRegistrationNumber" minOccurs="0"/>
            <xsd:element ref="com:FilingPlace" minOccurs="0"/>
            <xsd:element ref="com:ApplicantFileReference" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationLanguageCode" minOccurs="0"/>
            <xsd:element ref="com:SecondLanguageCode" minOccurs="0"/>
            <xsd:element ref="dgn:DesignTotalQuantity" minOccurs="0"/>
            <xsd:element ref="com:CorrespondenceLanguageCode" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationCurrentStatusCategory"
minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationCurrentStatusDate" minOccurs="0"/>
            <xsd:element ref="dgn:DesignatedCountryBag" minOccurs="0"/>
            <xsd:element ref="dgn:DesignBag"/>
            <xsd:element ref="dgn:ApplicantBag"/>
            <xsd:element ref="com:RepresentativeBag" minOccurs="0"/>
            <xsd:element ref="com:Authorization" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationEventBag" minOccurs="0"/>
            <xsd:element ref="com:DocumentIncludedBag" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationStatementBag" minOccurs="0"/>
            <xsd:element ref="com:PaymentBag" minOccurs="0"/>
            <xsd:element ref="com:ReimbursementBag" minOccurs="0"/>
            <xsd:element ref="com:SignatureBag" minOccurs="0"/>
            <xsd:choice>
                <xsd:element ref="com:ApplicationDate" minOccurs="0"/>
                <xsd:element ref="com:ApplicationDateTime"/>
            </xsd:choice>
            <xsd:element ref="com:CorrespondenceAddress" minOccurs="0"/>
            <xsd:element ref="com:BusinessEntityStatusCategory" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute ref="com:operationCategory"/>
        <xsd:attribute ref="com:st96Version" use="required"/>
        <xsd:attribute ref="com:ipoVersion"/>
    </xsd:complexType>
</xsd:schema>
```

Document level Component Type JSON Schema definition (designApplicationType_V5_0.json)

```
{
  "$id" : "designApplicationType_V5_0.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "designApplicationType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "operationCategory" : {
          "$ref" : "../../Common/operationCategory.json#/$defs/operationCategory"
        },
        "st96Version" : {
          "$ref" : "../../Common/st96Version.json#/$defs/st96Version"
        },
        "ipoVersion" : {
          "$ref" : "../../Common/ipoVersion.json#/$defs/ipoVersion"
        },
        "requestSoftware" : {
          "$ref" : "../../Common/requestSoftware.json#/$defs/requestSoftware"
        },
        "designApplicationFormName" : {
          "$ref" :
"../designApplicationFormName.json#/$defs/designApplicationFormName"
        },
        "requestExamination" : {
          "$ref" : "../../Common/requestExamination.json#/$defs/requestExamination"
        },
        "registrationOfficeCode" : {
          "$ref" :
"../../Common/registrationOfficeCode.json#/$defs/registrationOfficeCode"
        },
        "receivingOfficeCode" : {
          "$ref" : "../../Common/receivingOfficeCode.json#/$defs/receivingOfficeCode"
        },
        "receivingOfficeDate" : {
          "$ref" : "../../Common/receivingOfficeDate.json#/$defs/receivingOfficeDate"
        },
        "receiptNumber" : {
          "$ref" : "../receiptNumber.json#/$defs/receiptNumber"
        },
        "sealedDepositIndicator" : {
          "$ref" : "../sealedDepositIndicator.json#/$defs/sealedDepositIndicator"
        },
        "applicationNumber" : {
          "$ref" : "../../Common/applicationNumber.json#/$defs/applicationNumber"
        },
        "internationalRegistrationNumber" : {
          "$ref" :
"../../Common/internationalRegistrationNumber.json#/$defs/internationalRegistrationNum
ber"
        },
        "filingPlace" : {
          "$ref" : "../../Common/filingPlace.json#/$defs/filingPlace"
        },
        "applicantFileReference" : {
          "$ref" :
"../../Common/applicantFileReference.json#/$defs/applicantFileReference"
        },
        "designApplicationLanguageCode" : {
          "$ref" :
"../designApplicationLanguageCode.json#/$defs/designApplicationLanguageCode"
        },
        "secondLanguageCode" : {
          "$ref" : "../../Common/secondLanguageCode.json#/$defs/secondLanguageCode"
        },
        "designTotalQuantity" : {
          "$ref" : "../designTotalQuantity.json#/$defs/designTotalQuantity"
        },
        "correspondenceLanguageCode" : {
          "$ref" :
"../../Common/correspondenceLanguageCode.json#/$defs/correspondenceLanguageCode"
        },
        "designApplicationCurrentStatusCategory" : {
          "$ref" :
"../designApplicationCurrentStatusCategory.json#/$defs/designApplicationCurrentStatusC
ategory"
        },
        "designApplicationCurrentStatusDate" : {
          "$ref" :
"../designApplicationCurrentStatusDate.json#/$defs/designApplicationCurrentStatusDate"
        },
        "designatedCountryBag" : {
          "$ref" : "../designatedCountryBag.json#/$defs/designatedCountryBag"
        },
        "designBag" : {
          "$ref" : "../designBag.json#/$defs/designBag"
        },
        "applicantBag" : {
          "$ref" : "../applicantBag.json#/$defs/applicantBag"
        },
        "representativeBag" : {
          "$ref" : "../../Common/representativeBag.json#/$defs/representativeBag"
        },
        "authorization" : {
          "$ref" : "../../Common/authorization.json#/$defs/authorization"
        },
        "designApplicationEventBag" : {
          "$ref" :
"../designApplicationEventBag.json#/$defs/designApplicationEventBag"
        },
        "documentIncludedBag" : {
          "$ref" : "../../Common/documentIncludedBag.json#/$defs/documentIncludedBag"
        },
        "designApplicationStatementBag" : {
          "$ref" :
"../designApplicationStatementBag.json#/$defs/designApplicationStatementBag"
        },
        "paymentBag" : {
          "$ref" : "../../Common/paymentBag.json#/$defs/paymentBag"
        },
        "reimbursementBag" : {
          "$ref" : "../../Common/reimbursementBag.json#/$defs/reimbursementBag"
        },
        "signatureBag" : {
          "$ref" : "../../Common/signatureBag.json#/$defs/signatureBag"
        },
        "applicationDate" : {
          "$ref" : "../../Common/applicationDate.json#/$defs/applicationDate"
        },
        "applicationDateTime" : {
          "$ref" : "../../Common/applicationDateTime.json#/$defs/applicationDateTime"
        },
        "correspondenceAddress" : {
          "$ref" :
"../../Common/correspondenceAddress.json#/$defs/correspondenceAddress"
        },
        "businessEntityStatusCategory" : {
          "$ref" :
"../../Common/businessEntityStatusCategory.json#/$defs/businessEntityStatusCategory"
        }
      },
      "oneOf" : [ {
        "required" : [ "applicationDate" ]
      }, {
        "required" : [ "applicationDateTime" ]
      } ],
      "required" : [ "st96Version", "registrationOfficeCode", "designBag",
"applicantBag" ]
    }
  }
}
```

GLOBAL SCHEMA REFERENCING

[TR-05] The two conventional ways of global definitions used in XSDs, i.e., `xsd:import` and `xsd:include` MUST be transformed with “`$ref`” property in JSON schema irrespective of type used in XSD “`import`” or “`include`”.

For example:

XML Schema definition for xsd:import example (RelatedApplicationDate.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:dgn="http://www.wipo.int/standards/XMLSchema/ST96/Design"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../Common/DateType.xsd"/>
    <xsd:element name="RelatedApplicationDate" type="com:DateType">
        <xsd:annotation>
            <xsd:documentation>Application date of the related
application</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

JSON Schema definition for xsd:import example (relatedApplicationDate.json)

```
{
  "$id" : "relatedApplicationDate.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "relatedApplicationDate" : {
      "$ref" : "#/$defs/relatedApplicationDate"
    }
  },
  "required" : [ "relatedApplicationDate" ],
  "$defs" : {
    "relatedApplicationDate" : {
      "$ref" : "../Common/dateType.json#/$defs/dateType",
      "description" : "Description: Application date of the related application;
Version: V5_0"
    }
  }
}
```

XML Schema definition for xsd:include example (AffectedDesign.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:dgn="http://www.wipo.int/standards/XMLSchema/ST96/Design"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:include schemaLocation="AffectedDesignType.xsd"/>
    <xsd:element name="AffectedDesign" type="dgn:AffectedDesignType">
        <xsd:annotation>
            <xsd:documentation>Design affected by the decision, either all designs or
enumeration</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

JSON Schema definition for xsd:include example (affectedDesign.json)

```
{
  "$id" : "affectedDesign.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "affectedDesign" : {
      "$ref" : "#/$defs/affectedDesign"
    }
  },
  "required" : [ "affectedDesign" ],
  "$defs" : {
    "affectedDesign" : {
      "$ref" : "affectedDesignType.json#/$defs/affectedDesignType",
      "description" : "Description: Design affected by the decision, either all
designs or enumeration; Version: V5_0"
    }
  }
}
```

XSD COMPOSITORS

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`. `xsd:all` is not allowed in ST.96 per [SD-52] of WIPO ST.96 Annex I.

[TR-06] Object or array property to current component definition SHOULD be added according to "`maxOccurs`" value as guided below.

XSD	JSON	Notes
`xsd:sequence`	object	
`maxOccurs=1`	object	`minOccurs=1` will make object required
`maxOccurs=unbounded`	object or array	`minOccurs=1` will make object required
`xsd:choice`	object	
`maxOccurs=1`	object	`minOccurs=1` will make object required
`maxOccurs=unbounded`	object or array	`minOccurs=1` will make object required

For example:

XML Schema definition for xsd:sequence maxOccurs=1 example (AdditionalRemarkType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="languageCode.xsd"/>
    <xsd:complexType name="AdditionalRemarkType">
        <xsd:sequence>
            <xsd:element ref="com:P"/>
        </xsd:sequence>
        <xsd:attribute ref="com:languageCode"/>
    </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:sequence maxOccurs=1 example (additionalRemarkType.json)

```
{
  "$id" : "additionalRemarkType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "additionalRemarkType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "languageCode" : {
          "$ref" : "languageCode.json#/$defs/languageCode"
        },
        "p" : {
          "$ref" : "p.json#/$defs/p"
        }
      },
      "required" : [ "p" ]
    }
  }
}
```

XML Schema definition for xsd:sequence maxOccurs=unbounded example (InventionClaimBagType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:pat="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Patent"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="InventionNumber.xsd"/>
<xsd:include schemaLocation="ClaimNumber.xsd"/>
<xsd:include schemaLocation="ClaimNumberRange.xsd"/>
```

```
    <xsd:complexType name="InventionClaimBagType">
        <xsd:sequence maxOccurs="unbounded">
            <xsd:element ref="pat:InventionNumber"/>
            <xsd:choice maxOccurs="unbounded">
                <xsd:element ref="pat:ClaimNumber"/>
                <xsd:element ref="pat:ClaimNumberRange"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:sequence maxOccurs=unbounded example (inventionClaimBagType.json)

```
{
  "$id" : "inventionClaimBagType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "inventionClaimBagType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "inventionNumber" : {
          "type" : "array",
          "minItems" : 1,
          "items" : {
            "$ref" : "inventionNumber.json#/$defs/inventionNumber"
          }
        },
        "claimNumber" : {
          "anyOf" : [ {
            "$ref" : "claimNumber.json#/$defs/claimNumber"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "claimNumber.json#/$defs/claimNumber"
            }
          } ]
        },
        "claimNumberRange" : {
          "anyOf" : [ {
            "$ref" : "claimNumberRange.json#/$defs/claimNumberRange"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "claimNumberRange.json#/$defs/claimNumberRange"
            }
          } ]
        }
      },
      "anyOf" : [ {
        "required" : [ "claimNumber" ]
      }, {
        "required" : [ "claimNumberRange" ]
      } ],
      "required" : [ "inventionNumber" ]
    }
  }
}
```

XML Schema definition for xsd:choice maxOccurs=1 example (ChemicalFormulaeType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="Image.xsd"/>
<xsd:include schemaLocation="InlineFormula.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:include schemaLocation="chemicalFormulaeNumber.xsd"/>
<xsd:include schemaLocation="ExternalDocumentBag.xsd"/>
   <xsd:complexType name="ChemicalFormulaeType">
      <xsd:choice>
         <xsd:element ref="com:Image"/>
         <xsd:element ref="com:InlineFormula"/>
         <xsd:element ref="com:ExternalDocumentBag"/>
      </xsd:choice>
      <xsd:attribute ref="com:id"/>
      <xsd:attribute ref="com:chemicalFormulaeNumber"/>
   </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:choice maxOccurs=1 example (chemicalFormulaeType.json)

```
{
  "$id" : "chemicalFormulaeType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "chemicalFormulaeType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "id" : {
          "$ref" : "id.json#/$defs/id"
        },
        "chemicalFormulaeNumber" : {
          "$ref" : "chemicalFormulaeNumber.json#/$defs/chemicalFormulaeNumber"
        },
        "image" : {
          "$ref" : "image.json#/$defs/image"
        },
        "inlineFormula" : {
          "$ref" : "inlineFormula.json#/$defs/inlineFormula"
        },
        "externalDocumentBag" : {
          "$ref" : "externalDocumentBag.json#/$defs/externalDocumentBag"
        }
      },
      "oneOf" : [ {
        "required" : [ "image" ]
      }, {
        "required" : [ "inlineFormula" ]
      }, {
        "required" : [ "externalDocumentBag" ]
      } ]
    }
  }
}
```

XML Schema definition for xsd:choice maxOccurs=unbounded example (InventionClaimBagType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="Heading.xsd"/>
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:complexType name="ContentType">
       <xsd:choice maxOccurs="unbounded">
<xsd:element ref="com:Heading"/>
<xsd:element ref="com:P"/>
       </xsd:choice>
       <xsd:attribute ref="com:id"/>
   </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:choice maxOccurs=unbounded example (inventionClaimBagType.json)

```
{
  "$id" : "contentType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "contentType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "id" : {
          "$ref" : "id.json#/$defs/id"
        },
        "heading" : {
          "anyOf" : [ {
            "$ref" : "heading.json#/$defs/heading"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "heading.json#/$defs/heading"
            }
          } ]
        },
        "p" : {
          "anyOf" : [ {
            "$ref" : "p.json#/$defs/p"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "p.json#/$defs/p"
            }
          } ]
        }
      },
      "anyOf" : [ {
        "required" : [ "heading" ]
      }, {
        "required" : [ "p" ]
      } ]
    }
  }
```

ELEMENTS

Elements are the basic building blocks of an XML and must be transformed into a property in JSON.

[TR-07] Property with appropriate type MUST be added to current component definition according to the `maxOccurs` value as guided below.

XSD	JSON	Notes
`xsd:element`	Property	See Built-in data types transformation section above
`maxOccurs=1`	Property with XSD primitive type or custom type	`minOccurs=1` will make property required
`maxOccurs=unbounded`	Array of primitive type or custom type	`minOccurs=1` will make property required

For example:

XML Schema definition for xsd:element maxOccurs=1 example (DocumentTotalQuantity.xsd)

```
<?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="V5_0">
    <xsd:element name="DocumentTotalQuantity" type="xsd:nonNegativeInteger">
        <xsd:annotation>
            <xsd:documentation>Total number of documents available or provided.
</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

JSON Schema definition for xsd:element maxOccurs=1 example (documentTotalQuantity.json)

```
{
  "$id" : "documentTotalQuantity.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "additionalProperties" : false,
  "properties" : {
    "documentTotalQuantity" : {
      "$ref" : "#/$defs/documentTotalQuantity"
    }
  },
  "required" : [ "documentTotalQuantity" ],
  "$defs" : {
    "documentTotalQuantity" : {
      "type" : "integer",
      "minimum" : 0,
      "description" : "Description: Total number of documents available or provided.
; Version: V5_0"
    }
  }
}
```

XML Schema definition for xsd:element maxOccurs=unbounded example (IPOfficeCodeBagType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="IPOfficeCode.xsd"/>
    <xsd:complexType name="IPOfficeCodeBagType">
        <xsd:sequence>
            <xsd:element ref="com:IPOfficeCode" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:element maxOccurs=unbounded example (ipOfficeCodeBagType.json)

```
{
  "$id" : "ipOfficeCodeBagType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "ipOfficeCodeBagType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "ipOfficeCode" : {
          "type" : "array",
          "minItems" : 1,
          "items" : {
            "$ref" : "ipOfficeCode.json#/$defs/ipOfficeCode"
          }
        }
      },
      "required" : [ "ipOfficeCode" ]
    }
  }
}
```

ATTRIBUTES

Attributes are W3C Schema constructs associated with elements that provide further information regarding elements and must be transformed as a property of the current component in JSON.

[TR-08] Property with appropriate type MUST be added to current component definition as guided below:

XSD	JSON	Notes
`xsd:attribute`	Property of current component	See Section "Built-in data types transformation" section
`xsd:annotation`	“$description”	Please see Annotation section

For example:

XML Schema definition for xsd:attribute example (changeDateTime.xsd)

```
<?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="V5_0">
<xsd:attribute name="changeDateTime" type="xsd:dateTime">
        <xsd:annotation>
            <xsd:documentation>Date and time of change</xsd:documentation>
        </xsd:annotation>
    </xsd:attribute>
</xsd:schema>
```

JSON Schema definition for xsd:attribute example (changeDateTime.json)

```
{
  "$id" : "changeDateTime.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "additionalProperties" : false,
  "properties" : {
    "changeDateTime" : {
      "$ref" : "#/$defs/changeDateTime"
    }
  },
  "required" : [ "changeDateTime" ],
  "$defs" : {
    "changeDateTime" : {
      "format" : "date-time",
      "type" : "string",
      "description" : "Description: Date and time of change; Version: V5_0"
    }
  }
}
```

SIMPLETYPE

[TR-09] A property with an appropriate type SHOULD be added to current component definition as guided below.

XSD	**JSON**	**Notes**
`xsd:simpleType`	object	See Built-in data types transformation section
`    xsd:union`	object - "`anyOf`"	See Union section
`    xsd:restriction`	object	See Restriction section
`        xsd:enumeration`	enum	See Enumeration section

COMPLEXTYPE

[TR-10] A property with an appropriate type SHOULD be added to current component definition as guided below.

XSD	**JSON**	**Notes**
`Xsd:complexType`	object	See Built-in data types transformation section
`    xsd:simpleContent`	object	See Simple Content Section
`    xsd:complexContent`	object	See Complex Content section
`    xsd:sequence`	object	See XSD Compositors section
`    xsd:choice`	object/array	See XSD Compositors section
`    @mixed`	object	See Mixed Content section

Simple Content

[TR-11] A property with an appropriate type SHOULD be added to current component definition as guided below.

XSD	**JSON**	**Notes**
`xsd:simpleContent`	object	
`    xsd:extension`	object	See Extension section
`    xsd:restriction`	object	See Restriction section

Complex Content

[TR-12] A property with an appropriate type SHOULD be added to current component definition as guided below.

XSD	**JSON**	**Notes**
`xsd:complexContent`	object	
`    xsd:extension`	object	See Extension section
`    xsd:restriction`	object	See Restriction section

Mixed Content

[TR-13] A property with an appropriate type SHOULD be added to current component definition as guided below.

XSD	**JSON**	**Notes**
`xsd:complexType`	object	
`    @mixed`	object	

For example:

XML Schema definition for xsd:complexType\@mixed=true example (CrossReferenceType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="PhraseType.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:include schemaLocation="idrefs.xsd"/>
<xsd:include schemaLocation="extRef.xsd"/>
<xsd:include schemaLocation="crossReferenceCategory.xsd"/>
<xsd:include schemaLocation="sourceURI.xsd"/>
<xsd:include schemaLocation="sourceSystemName.xsd"/>
<xsd:include schemaLocation="sourceSystemIdentifier.xsd"/>
    <xsd:complexType name="CrossReferenceType" mixed="true">
        <xsd:complexContent>
            <xsd:extension base="com:PhraseType">
                <xsd:attribute ref="com:id"/>
                <xsd:attribute ref="com:idrefs"/>
                <xsd:attribute ref="com:extRef"/>
                <xsd:attribute ref="com:crossReferenceCategory" use="required"/>
                <xsd:attribute ref="com:sourceURI"/>
                <xsd:attribute ref="com:sourceSystemName"/>
                <xsd:attribute ref="com:sourceSystemIdentifier"/>
                </xsd:extension>
         </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:complexType\@mixed=true example (crossReferenceType.xsd)

```
{
  "$id" : "crossReferenceType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "crossReferenceType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "phraseType" : {
          "$ref" : "phraseType.json#/$defs/phraseType"
        },
        "id" : {
          "$ref" : "id.json#/$defs/id"
        },
        "idrefs" : {
          "$ref" : "idrefs.json#/$defs/idrefs"
        },
        "extRef" : {
          "$ref" : "extRef.json#/$defs/extRef"
        },
        "crossReferenceCategory" : {
          "$ref" : "crossReferenceCategory.json#/$defs/crossReferenceCategory"
        },
        "sourceURI" : {
          "$ref" : "sourceURI.json#/$defs/sourceURI"
        },
        "sourceSystemName" : {
          "$ref" : "sourceSystemName.json#/$defs/sourceSystemName"
        },
        "sourceSystemIdentifier" : {
          "$ref" : "sourceSystemIdentifier.json#/$defs/sourceSystemIdentifier"
        }
      },
      "required" : [ "crossReferenceCategory" ]
    }
  }
}
```

ANNOTATION

There are two types of annotation used in WIPO Standard ST.96, i.e., `xsd:appinfo` and `xsd:documentation`.

`xsd:appinfo`

The `xsd:appinfo` element specifies information to be used by the application. This element must go within an annotation element. ST.96 uses `xsd:appinfo` in all the document-level XSDs containing the following schema elements:

– `com:SchemaCreatedDate`

– `com:SchemaLastModifiedDate`

– `com:SchemaContactPoint`

– `com:SchemaReleaseNoteURL`

[TR-14] Information provided under `<xsd:appinfo>` SHOULD be transferred to the "description" value in a JSON Schema.

For example:

Component XML Schema definition (ApplicationBodyType.xsd)

```
      <xsd:annotation>
            <xsd:appinfo>
                  <com:SchemaCreatedDate>2012-07-13</com:SchemaCreatedDate>
                  <com:SchemaLastModifiedDate>2021-10-01</com:SchemaLastModifiedDate>

      <com:SchemaContactPoint>xml.standards@wipo.int</com:SchemaContactPoint>

      <com:SchemaReleaseNoteURL>http://www.wipo.int/standards/XMLSchema/ST96/V5_0/Relea
seNotes.pdf</com:SchemaReleaseNoteURL>
            </xsd:appinfo>
      </xsd:annotation>
      <xsd:include schemaLocation="ApplicationBodyType_V5_0.xsd"/>
      <xsd:element name="ApplicationBody" type="pat:ApplicationBodyType">
            <xsd:annotation>
                  <xsd:documentation>Body of a patent application</xsd:documentation>

            </xsd:annotation>...
```

JSON Schema definition for xsd:annotation example (applicationBodyType.json)

```
"$defs" : {
      "applicationBody" : {
            "$ref" : "applicationBodyType_V5_0.json#/$defs/applicationBodyType",
                "description" : "Description: Body of a patent application; Version:
      V5_0; SchemaCreatedDate: 2012-07-13; SchemaLastModifiedDate: 2021-10-01;
      SchemaContactPoint: xml.standards@wipo.int;
      SchemaReleaseNoteURL: http://www.wipo.int/standards/XMLSchema/ST96/V5_0/ReleaseNo
      tes.pdf"
              }

      }
```

`xsd:documentation`

In ST.96, this specifies information to be read or used by users within an annotation element. According to ST.96:

> *"[SD-58], all schemas SHOULD include schema construct documentation using the xsd:documentation element."*

[TR-15] Documentation information and version information SHOULD be transferred to "`description`" value in JSON Schema.

For example:

Component XML Schema definition (AbstractNumber.xsd)

```
    <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="V5_0">
      <xsd:element name="AbstractNumber" type="xsd:string">
            <xsd:annotation>
                  <xsd:documentation>Number assigned to an abstract published without
    the full document in a collection of abstracts. This collection can be a journal,
    conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
    Abstracts), etc.</xsd:documentation>
            </xsd:annotation>
      </xsd:element>
    </xsd:schema>
```

JSON Schema definition for xsd:documentation example (abstractNumber.json):

```
{
  "$id" : "abstractNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "additionalProperties" : false,
  "properties" : {
    "abstractNumber" : {
      "$ref" : "#/$defs/abstractNumber"
    }
  },
  "required" : [ "abstractNumber" ],
  "$defs" : {
    "abstractNumber" : {
      "type" : "string",
      "description" : "Description: Number assigned to an abstract published without
the full document in a collection of abstracts. This collection can be a journal,
conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
Abstracts), etc.; Version: V5_0"
    }
  }
}
```

UNION

[TR-16] Property with appropriate type SHOULD be added to current component definition as guided in the table below.

XSD	JSON	Notes
`xsd:union`	object - “`anyOf`”	See Built-in data types transformation section

For example:

XML Schema definition for xsd:simpleType\xsd:union example (DocumentNameType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="DocumentNameCategoryType.xsd"/>
<xsd:simpleType name="DocumentNameType">
   <xsd:union memberTypes="xsd:string com:DocumentNameCategoryType"/>
```

```
    </xsd:simpleType>
</xsd:schema>
```

JSON Schema definition for xsd:simpleType\xsd:union example (documentNameType.json)

```
{
  "$id" : "documentNameType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "documentNameType" : {
      "description" : "Version: V5_0",
      "anyOf" : [ {
        "type" : "string"
      }, {
        "$ref" : "documentNameCategoryType.json#/$defs/documentNameCategoryType"
      } ]
    }
  }
}
```

EXTENSION

[TR-17] An object SHOULD be added to current component definition as guided in the table below:

XSD	JSON	Notes
`xsd:extension`	object - “`anyOf`”	See Built-in data types transformation section

For example:

XML Schema definition for xsd:extension example (AmountType.xsd)

```
<?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="V5_0">
<xsd:include schemaLocation="currencyCode.xsd"/>
<xsd:complexType name="AmountType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute ref="com:currencyCode"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```

JSON Schema definition for xsd:extension example (amountType.json)

```
{
  "$id" : "amountType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "amountType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "$" : {
          "type" : "number"
        },
        "currencyCode" : {
          "$ref" : "currencyCode.json#/$defs/currencyCode"
        }
      }
    }
  }
}
```

RESTRICTION

[TR-18] Object SHOULD be added to current component definition as guided in the table below.

XSD	JSON	Notes
`xsd:restriction`	object	
`xsd:pattern`	pattern	Please see Pattern section
`xsd:enumeration`	enum	Please see Enumeration section

ENUMERATION

[TR-19] The `xsd:enumeration` MUST be transformed to "`enum`" property in current component definition of JSON Schema.

XML Schema definition for xsd:simpleType\xsd:restriction\xsd:enumeration example (BusinessEntityStatusCategoryType.xsd)

```
<?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="V5_0">
    <xsd:simpleType name="BusinessEntityStatusCategoryType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="Undiscounted">
                <xsd:annotation>
                    <xsd:documentation>Undiscounted entity</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="Small">
                <xsd:annotation>
                    <xsd:documentation>Small entity discount</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="Micro">
                <xsd:annotation>
                    <xsd:documentation>Micro entity discount</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

JSON Schema definition for xsd:simpleType\xsd:restriction\xsd:enumeration example (businessEntityStatusCategoryType.json)

```
{
  "$id" : "businessEntityStatusCategoryType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "businessEntityStatusCategoryType" : {
      "description" : "Version: V5_0; Undiscounted: Undiscounted entity; Small: Small
entity discount; Micro: Micro entity discount",
      "type" : "string",
      "enum" : [ "Undiscounted", "Small", "Micro" ]
    }
  }
}
```

CONSTRAINING FACETS

W3C Schemas use the constraining facets listed in the table below. It should be noted that `xsd:minInclusive`, `xsd:maxInclusive`, `xsd:minExclusive`, `xsd:maxExclusive`, and `xsd:minLength` are not used in ST.96. Consequently those unused facets are not provided in Annex I.

[TR-20] XSD constraining facets MUST be transformed to the corresponding JSON keywords along with appropriate minimum or maximum length as guided in Table below.

Table 3 Transformation of XSD constraining facets (X is the constraint numeral value)

XSD constraining facet	JSON Schema equivalent
`<xsd:minLength value="X" />`	`{ "minLength": X }`
`<xsd:maxLength value="X" />`	`{ "maxLength": X }`
`<xsd:length value="X" />`	`{ "minLength": X, "maxLength": X }`
`<xsd:pattern value="X" />`	`{ "pattern": "X" }`
`<xsd:minExclusive value="X" />`	`{ "minimum": X, "exclusiveMinimum": true }`
`<xsd:maxExclusive value="X" />`	`{ "maximum": X, "exclusiveMaxmimum": true }`
`<xsd:minInclusive value="X" />`	`{ "minimum": X, "exclusiveMinimum": false }`
`<xsd:maxInclusive value="X" />`	`{ "maximum": X, "exclusiveMaximum": false }`

For example:

XML Schema definition for xsd:length example (ClassType.xsd)

```
<?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" version="V5_0">
    <xsd:simpleType name="ClassType">
        <xsd:restriction base="xsd:token">
            <xsd:length value="2"/>
            <xsd:pattern value="[0-9][1-9]|[1-9][0-9]"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

JSON Schema definition for xsd:length example (classType.json)

```
{
  "$id" : "classType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "classType" : {
      "description" : "Version: V5_0",
      "type" : "string",
      "maxLength" : 2,
      "pattern" : "[0-9][1-9]|[1-9][0-9]"
    }
  }
}
```

Pattern

[TR-21] `xsd:pattern` MUST be transformed to a “`pattern`” property in current component definition.

For example:

XML Schema definition for xsd:pattern example (WIPONotificationNumberType.xsd)

```
<?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="V5_0">
<xsd:simpleType name="WIPONotificationNumberType">
       <xsd:restriction base="xsd:token">
           <xsd:pattern value="[A-Z]{3}[0-9]{6}"/>
       </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

JSON Schema definition for xsd:pattern example (wipoNotificationNumberType.json)

```
{
  "$id" : "wipoNotificationNumberType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "wipoNotificationNumberType" : {
      "description" : "Version: V5_0",
      "type" : "string",
      "pattern" : "[A-Z]{3}[0-9]{6}"
    }
  }
}
```

GROUP

The `xsd:group` is not used in ST.96, but is used in External Standards which are referred to by ST.96 XSDs. Further information on external Standards is available in Section “Transformation of external XSD dependencies” below.

[TR-22] The object type SHOUD be added to current component definition.

XSD	JSON	Notes
`xsd:group`	object	

For example:

XML Schema definition for xsd:group example (fragment from FlattenedMathML3.json)

```
<xs:group name="anyElement">
       <xs:choice>
           <xs:any namespace="##other" processContents="skip"/>
           <xs:any namespace="##local" processContents="skip"/>
       </xs:choice>
   </xs:group>
```

JSON Schema definition for xsd:group example (fragment from flattenedMathML3.json)

```
"anyElement" : {
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "other_SKIP" : {
          "patternProperties" : {
            "^@\\w+$" : {
              "type" : [ "string", "number", "boolean" ]
            }
          },
          "additionalProperties" : false,
          "type" : [ "object", "string", "number", "boolean" ],
          "properties" : {
            "$" : {
              "type" : [ "string", "number", "boolean" ]
            }
          }
        },
        "local_SKIP" : {
          "patternProperties" : {
            "^@\\w+$" : {
              "type" : [ "string", "number", "boolean" ]
            }
          },
          "additionalProperties" : false,
          "type" : [ "object", "string", "number", "boolean" ],
          "properties" : {
            "$" : {
              "type" : [ "string", "number", "boolean" ]
            }
          }
        }
      },
      "oneOf" : [ {
        "required" : [ "other_SKIP" ]
      }, {
        "required" : [ "local_SKIP" ]
      } ]
    }
```

TRANSFORMATION OF EXTERNAL XSD DEPENDENCIES

WIPO Standard ST.96 refers to the following industry-standard schemas instead of redefining them in ST.96:

– MathML version 3[3] (`FlattenedMathML3.xsd`); and

– OASIS Table Schema[4] (`OASISTable_V1_0.xsd`).

At the time of developing this Standard, these external Standards do not provide equivalent JSON Schemas. However, recognizing that ST.96-based JSON Schemas are not possible without ~~finding~~ equivalent JSON Schemas, as a programmatic solution, these two external XSDs are transformed into reasonably stable equivalent JSON Schemas using the Transformation Tool provided in the Appendix to Annex I of this Standard and the transformed JSON Schemas are included in the set of JSON Schemas as Annex II.

[Appendix to Annex I of the proposed Standard follows]

[3] http://www.w3.org/TR/MathML3
[4] http://www.oasisopen.org/docbook/xmlschema/1.0b1/calstbl.xsd

APPENDIX

XSD TO JSON SCHEMA TRANSFORMATION TOOL

This Appendix to Annex I contains the tool for transformation of XML Schemas to JSON Schemas which is a Java library that helps IPOs transform a provided WIPO ST.96-based XSD to its JSON Schema equivalent, according to the transformation rules provided in Annex I. This Transformation Tool is provided here as part of the Standard such that IPOs may also utilize this tool to transform their own customized ST.96 XSDs into JSON Schema.

Requirements

Java Runtime Environment 1.8 (or higher)

Usage

~~C:\>~~#Provides help options
```
C:\>java -jar xsd2JsonSchema.jar -help
```

~~C:\>~~#Transform a single file (XSD)
```
C:\>java -jar xsd2JsonSchema.jar -f "C:\XSD_Folder~~_Path~~\Common\AbstractNumber.xsd"
```

~~C:\>~~#Transform a single file (XSD) and the included/imported schemas
```
C:\>java -jar xsd2JsonSchema.jar -f -r "C:\XSD_Folder_Path\Common\AbstractNumber.xsd"
```

~~C:\>~~#Transform all the schemas of the provided directory (the recursive flag is not available with this option)
```
C:\>java -jar xsd2JsonSchema.jar -d "C:\XSD_Folder_Path\Common"
```

Download Executable Jar

The tool can be downloaded in the Appendix of Annex I at the following link:
https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixi.zip

[Annex II of the proposed Standard follows]

ANNEX II

JSON SCHEMA

[Editorial Notes: The transformation rules of Annex I are also applied to the transformed exernal XML standards, i.e., MathML and OASIS Table. Those tag names were kept as is just like ST.96 XSD names but it applies LCC rule to all names, e.g., MathExpression to mathExpression in MathML, which is not intended. All tag names of the external XML standards should be kept as they are. Therefore the changed tag names will be restored for the publication of the new standard.]

Annex II provides a full-set of JSON Schemas which correspond to the WIPO Standard ST.96 XSDs, version 5.0. These JSON Schemas were automatically produced using the transformation tool provided in the Appendix to Annex I according to the transformation rules and guidelines defined in Annex I. It should be noted that this is a one-way transformation process, i.e., from XSD to JSON. This set of JSON Schemas include the transformed JSON Schemas of the external XML standards, i.e., MathML and OASIS Table. However, the original tag names of XSD components of the external XML standards are kept as they are.

The JSON Schemas can be downloaded in the Appendix of Annex II at the following link:
https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixii.zip

[Annex III of the proposed Standard follows]

ANNEX III

JSON EXAMPLE INSTANCES

Annex III provides examples of JSON instances which are derived from WIPO Standard ST.96 XSDs in order to support IPOs in production of similar instances. Each of these instances should validate against the relevant JSON Schema provided in Annex II.

The following example instances correspond to some of document-level components, which are provided in Annex VII of WIPO ST.96, and do not reflect real data:

– **`patentPublication.json`**: this document-level component is used to capture details of the publication of a patent application. This example instance can be downloaded here: *patentPublication.json*

– **`trademarkApplication.json`**: this document-level component is used to capture information related to a filed trademark application. This example instance can be downloaded here: trademarkApplication.json

– **`designApplication.json`**: this document-level component is used to capture information related to a filed industrial design application. This example instance can be downloaded here: designApplication.json

[Note: downloadable links for these example instances will be provided above after publication of the current draft standard]

The JSON example instances can be downloaded in the Appendix of Annex III at the following link: https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixiii.zip

[Annex IV of the proposed Standard follows]

ANNEX IV

LIST OF ACRONYMS AND ABBREVIATIONS

Acronyms and abbreviations appearing at the beginning of an object type and property name MUST be in lower case e.g., "pre", "bioDeposit". If an acronym is provided all in capitals at the beginning of a name, all characters must be lower case, e.g., “idref” and “wipo” in the property name of “wipoST3Code”. Otherwise all values of an enumeration, acronyms and abbreviation values MUST appear in upper case as listed below.

The acronyms/abbreviations below SHOULD NOT be considered in the context of the Language codes, Currency codes, Office codes and Country codes, which are listed in WIPO Standard ST.96, where there may be duplicate values. These codes are based on ISO 639-1 language codes, ISO 4217 currency codes, WIPO ST.3 codes and ISO 3166-1 country codes respectively.

AF	Authority File
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
EIDR	Entertainment Identifier Registry
ExtRef	References that are external to the current XML document
GI	Geographical Indication
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
IB	International Bureau
ID	Identifier for system identification
IDREF	Identifier Reference
IDREFS	Identifier References
IGES	Initial Graphic Exchange Specification
IGO	Intergovernmental organization the legislation of which provides for the protection of intellectual property rights or which organizations are acting in the framework of a treaty in the field of intellectual property
INID	Internationally agreed Numbers for the Identification of (bibliographic) Data
Ins	Inserted text
IP	Intellectual Property
IPC	International Patent Classification
IPCR	International Patent Classification Reform
IPO	Intellectual Property Office
IPR	Intellectual Property Right
ISMN	International Standard Music Number
ISNI	International Standard Name Identifier
ISO	International Organization for Standardization
JSON	Javascript Object Notation
LCC	Lower Camel Case
LI	List Item
LOR	License Of Right
MathML	Description Mathematical Markup Language

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
NUTS	Nomenclature of Territorial Units for Statistics
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
SEQL	Sequence listing
SOC	Society Code
SPC	Supplementary Protection Certificate
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
TISA	CISAC Territory Information System Code - Alphanumeric
TISN	CISAC Territory Information System Code - Numeric
TSG	Traditional Specialties Guaranteed
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

[Annex V of the proposed Standard follows]

ANNEX V

REPRESENTATION TERMS

Term	Definition	Data Type
Amount	A monetary value.	`number`
Category	A specifically defined division or subset in a system of classification in which all items share the same concept of taxonomy.	`string`
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.	`string`
Date	The notion of a specific point in time, expressed by year, month, and day.	`string`, with keyword `"format": "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.	`string`
Indicator	A signal of the presence, absence, or requirement of something. Boolean values are either `true` or `false` (without quotes). These values are case sensitive.	`boolean or string`
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.	`number`
Name	The designation of an object expressed in a word or phrase.	`String`
Number	A string of numeral or alphanumeric characters expressing label, value, quantity or identification.	`integer`, `number`, or `string`
Percent	A number which represents a part of a whole, which will be divided by 100.	`number`
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 `Integer` should be used for countable components which do not need unit information.	`number` with keyword `"minimum": 0 or integer`
Rate	A quantity or amount measured in relation to another quantity or amount.	`number`
Text	A formatted or an unformatted character string, generally in the form of words (includes: Abbreviation, Comments.)	`string`
Time	A designation of a specified chronological point within a period.	`string`, with keyword `"format": "time"`
DateTime	The captured date and time of an event when it occurs.	`string`, with keyword `"format": "date-time"`
URI	The Uniform Resource Identifier that identifies where the file is located.	`string`, with keyword `"format":` and values `"uri" or "uri-reference"`

[End of Annex V of the proposed Standard and of the proposed Standard]

[Конец приложения и документа]