WIPO STANDARD ST.97

Version 2.0

RECOMMENDATION FOR PROCESSING INTELLECTUAL PROPERTY DATA USING JSON

*Revision approved by the API Task Force of the
Committee of WIPO Standards (CWS) for publication on July 15, 2026*

TABLE OF CONTENTS

INTRODUCTION 3
DEFINITIONS AND TERMINOLOGY 3
GENERAL NOTATIONS 4
Rule Identifiers 4
Sample JSON Data Structure 5
SCOPE OF THE STANDARD 5
JSON GENERAL DESIGN RULES AND CONVENTIONS 5
Overview 5
JSON Naming Conventions 6
JSON SCHEMA DESIGN RULES 7
Overview 7
Modularity 8
Documentation 16
Schema file naming convention 17
JSON Schema Properties Structuring 18
JSON SCHEMA CONSTRUCTS DESIGN RULES 19
Overview 19
Properties 19
Definitions 19
Type Definitions 20
JSON Primitive Type 20
Code Lists 20
Arrays 21
Objects 21
JSON SCHEMA IDENTIFIERS 21
Overview 21
JSON INSTANCE DESIGN RULES 21
Order of Properties 22
JSON instance validation 22

REFERENCES 22
WIPO Standards 22
Standards and Industry Specifications 22

ANNEXES

The Annexes are available at: http://www.wipo.int/standards/en/st97/v2-0/

Annex I – Transformation rules form Standard ST.96 XSD to JSON SCHEMA and Guidelines for use

Annex II – JSON SCHEMA

Annex III – JSON example instances

Annex IV – List of acronyms and abbreviations

Annex V – Representation terms

Annex VI – IP Data Dictionary

WIPO STANDARD ST.97

Version 2.0

RECOMMENDATION FOR PROCESSING INTELLECTUAL PROPERTY DATA USING JSON

Revision approved by the API Task Force of the Committee of WIPO Standards (CWS) for publication on July 15, 2026

INTRODUCTION

1. 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 includes rules to transform WIPO Standard ST.96 eXtensible Markup Language (XML) Schemas (XSDs) to JSON Schemas that meet the aforementioned recommendations, as the JSON Schemas are derived from WIPO Standard ST.96.

2. This Standard is intended to:

- Deliver well-structured JSON Schemas and Instances;
- Provide guidance on designing and developing IP JSON data best practices;
- Recommend design principles for extending the provided JSON Schemas or creating new conformant JSON Schemas; and
- Improve data exchange efficiency by promoting the reuse of JSON Schemas and Instances among Intellectual Property Offices (IPOs), as well as data made available to the public.

DEFINITIONS AND TERMINOLOGY

3. For purposes of the Standard, the following definition are used:

- The term “JSON resources” in this Standard refers to any of the components used to create and operate a JSON implementation compliant to this Standard;
- The terms “JSON”, “JSON text” , “JSON value” “array”, “element”, “object”, “object type”, “property”, “member”, “property name”, “property value”, “property type”, “keyword”, “number”, “string”, “boolean”, “true”, “false”, “null” 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;
- The term “patent” includes such industrial property rights as patents for inventions, plant patents, design patents, inventors’ certificates, utility certificates, utility models, patents of addition, inventors’ certificates of addition and utility certificates of addition;
- The term “trademark” means trademark, service mark or another type of distinguishing representation of mark according to the definition of the mark in the legislation concerned, including but not limited to collective mark, certification mark or guarantee mark;

[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

- The term “industrial design” includes two-dimensional and three-dimensional features of shape and surface of objects, and thus covers both concepts of “designs” and “models” where a distinction is made between the former and the latter; the term “industrial designs” does not include design patents;

- The term “geographical indication” means a sign used on products that have a specific geographical origin and possess qualities or a reputation that are due to that origin; and

- The term “copyright’ includes rights that creators have over their literary and artistic works and the expression “copyright orphan work” includes works that are protected by copyright, but the author cannot be identified or found.

4. The keywords “MUST”, “MUST NOT”, "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL", when they appear capitalized in this Standard are to be interpreted as described below only when they appear in all capitals. Non-capitalized forms of these words are used in the regular English sense.

- MUST: This word, or the terms “REQUIRED” or “SHALL”, means that the definition is an absolute requirement of the specification;

- MUST NOT: This phrase, or the phrase “SHALL NOT”, means that the definition is an absolute prohibition of the specification;

- SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course;

- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label; and

- MAY: This word, or the adjective “OPTIONAL”, means that an item is truly optional. An implementation that does not include a particular option MUST be prepared to interoperate with another implementation that does include the option, though perhaps with reduced functionality. In the same vein, an implementation that does include a particular option MUST be prepared to interoperate with another implementation that does not include the option (except, of course, for the feature the option provides).

GENERAL NOTATIONS

5. The following notations are used throughout this Standard:

- <>: Indicates a placeholder descriptive term that, in implementation, will be replaced with a specific instance value. This convention does not apply where it appears as part of Schema syntax;

- “ ”: Indicates that the text included in quotes must be used verbatim in implementation. This convention does not apply where it appears as part of Schema syntax;

- { }: Indicates that the items are optional in implementation. This convention does not apply where it appears as part of Schema syntax; and

- `Courier New` font: Indicates JSON and XML keywords, component names and codes.

Rule Identifiers

6. 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

  (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”.

Sample JSON Data Structure

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

SCOPE OF THE STANDARD

8. 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.

9. 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).

10. The Design Rules and Conventions (DRC) contained within this document provide a comprehensive set of rules for the creation of JSON Schemas and Instances used for all types of IP information. The transformation rules are provided to guidance to IPOs on how transform their customized ST.96 Schemas to JSON Schema equivalents.

11. This Standard includes the following Annexes:

- Annex I: Transformation rules form Standard ST.96 XSD to JSON SCHEMA and Guidelines for use;
- Annex II: JSON Schemas[2];
- Annex III: JSON example instances;
- Annex IV: List of acronyms and abbreviations;
- Annex V: Representational terms; and
- Annex VI: IP data dictionary

12. This Standard excludes the following:

(a) Software architectural concerns; and

(b) Implementation languages.

13. All objects, excluding data type objects, and enumeration values defined by this Standard MUST be listed in the IP Data Dictionary, i.e., Annex VI of the Standard.

14. Every version of Annex II MUST be released in its entirety including updated and unmodified JSON Schema components.

JSON GENERAL DESIGN RULES AND CONVENTIONS

Overview

15. This section contains general, high-level JSON design rules and conventions 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 conventions, 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

[2] The transformed JSON Schemas have generally the same component names and the data structure as defined in the corresponding version of the ST.96, Annex III. However several manual changes are made to some JSON Schemas necessary to implement difference in the XML and JSON technologies. Further details on the manually changed schema are provided to the release notes of the respective ST.97 version.

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.97 or the compatibility with ST.96 is not necessarily to be considered.

JSON Naming Conventions

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

17. These JSON naming conventions set out in this Section are based on the guidelines and principles described in document ISO 11179 Part 5[3] - Naming and Identification Principles. The object names and property names are composed 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 the 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. Use of abbreviations and acronyms is encouraged to reduce component name length. Any new abbreviations or acronyms MUST be added to Annex IV.

[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-24] Acronyms and abbreviations appearing at the beginning of a component name MUST be in lower camel case, e.g., "`pre`", "`bioDeposit`". If an acronym is provided all in capitals and it appears at the beginning of a component 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 appear as listed in Annex IV.

[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`.

[3] https://www.iso.org/standard/60341.html

[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.

[JGD-23] *Deleted*

JSON SCHEMA DESIGN RULES

Overview

18. 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[4] 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, , and JSON Schema Validation, version 2020-12.

[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"`

19. The JSON schema must be encoded using UTF-8 for maximum interoperability.

[4] https://json-schema.org/draft/2020-12/schema

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

Modularity

20. 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.

21. 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: V2_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" :     Andorra: AD; United Arab Emirates: AE; Afghanistan:
AF; Antigua And Barbuda: AG; Anguilla: AI; Albania: AL; Armenia: AM; Angola: AO;
African Regional Intellectual Property Organization (ARIPO): AP; Argentina: AR;
Austria: AT; Australia: AU; Aruba: AW; Azerbaijan: AZ; Bosnia and Herzegovina: BA;
Barbados: BB; Bangladesh: BD; Belgium: BE; Burkina Faso: BF; Bulgaria: BG; Bahrain:
BH; Burundi: BI; Benin: BJ; Bermuda: BM; Brunei Darussalam: BN; Bolivia
(Plurinational State of): BO; Bonaire, Sint Eustatius and Saba: BQ; Brazil: BR;
```

```
Bahamas: BS; Bhutan: BT; Bouvet Island: BV; Botswana: BW; Benelux Office for
Intellectual Property (BOIP): BX; Belarus: BY; Belize: BZ; Canada: CA; Democratic
Republic of the Congo: CD; Central African Republic: CF; Congo: CG; Switzerland:
CH; Côte d'Ivoire: CI; Cook Islands: CK; Chile: CL; Cameroon: CM; China: CN;
Colombia: CO; Costa Rica: CR; Cuba: CU; Cabo Verde: CV; Curaçao: CW; Cyprus: CY;
Czechia: CZ; Germany: DE; Djibouti: DJ; Denmark: DK; Dominica: DM; Dominican
Republic: DO; Algeria: DZ; Eurasian Patent Organization (EAPO): EA; Ecuador: EC;
Estonia: EE; Egypt: EG; Western Sahara: EH; European Union Intellectual Property
Office (EUIPO): EM; European Patent Office (EPO): EP; Eritrea: ER; Spain: ES;
Ethiopia: ET; European Union: EU; Finland: FI; Fiji: FJ; Falkland Islands
(Malvinas): FK; Micronesia (Federated states of): FM; Faroe Islands: FO; France:
FR; Gabon: GA; United Kingdom: GB; Patent Office of the Cooperation Council for the
Arab States of the Gulf (GCC Patent Office): GC; Grenada: GD; Georgia: GE;
Guernsey: GG; Ghana: GH; Gibraltar: GI; Greenland: GL; Gambia: GM; Guinea: GN;
Equatorial Guinea: GQ; Greece: GR; South Georgia and South Sandwich Islands: GS;
Guatemala: GT; Guinea-Bissau: GW; Guyana: GY; Hong Kong, China: HK; Honduras: HN;
Croatia: HR; Haiti: HT; Hungary: HU; International Bureau of the World Intellectual
Property Organization (WIPO): IB; Indonesia: ID; Ireland: IE; Israel: IL; Isle of
Man: IM; India: IN; Iraq: IQ; Iran (Islamic Republic of): IR; Iceland: IS; Italy:
IT; Jersey: JE; Jamaica: JM; Jordan: JO; Japan: JP; Kenya: KE; Kyrgyzstan: KG;
Cambodia: KH; Kiribati: KI; Comoros: KM; Saint Kitts and Nevis: KN; Democratic
People's Republic of Korea: KP; Republic of Korea: KR; Kuwait: KW; Cayman Islands:
KY; Kazakhstan: KZ; Lao People's Democratic Republic: LA; Lebanon: LB; Saint Lucia:
LC; Liechtenstein: LI; Sri Lanka: LK; Liberia: LR; Lesotho: LS; Lithuania: LT;
Luxembourg: LU; Latvia: LV; Libya: LY; Morocco: MA; Monaco: MC; Republic of
Moldova: MD; Montenegro: ME; Madagascar: MG; Marshall Islands: MH; North Macedonia:
MK; Mali: ML; Myanmar: MM; Mongolia: MN; Macao, China: MO; Northern Mariana
Islands: MP; Mauritania: MR; Montserrat: MS; Malta: MT; Mauritius: MU; Maldives:
MV; Malawi: MW; Mexico: MX; Malaysia: MY; Mozambique: MZ; Namibia: NA; Niger: NE;
Nigeria: NG; Nicaragua: NI; Netherlands (Kingdom of the): NL; Norway: NO; Nepal:
NP; Nauru: NR; Niue: NU; New Zealand: NZ; African Intellectual Property
Organization (OAPI): OA; Oman: OM; Panama: PA; Peru: PE; Papua New Guinea: PG;
Philippines: PH; Pakistan: PK; Poland: PL; Portugal: PT; Palau: PW; Paraguay: PY;
Qatar: QA; Community Plant Variety Office (European Union) (CPVO): QZ; Romania: RO;
Serbia: RS; Russian Federation: RU; Rwanda: RW; Saudi Arabia: SA; Solomon Islands:
SB; Seychelles: SC; Sudan: SD; Sweden: SE; Singapore: SG; Saint Helena, Ascension
and Tristan da Cunha: SH; Slovenia: SI; Slovakia: SK; Sierra Leone: SL; San Marino:
SM; Senegal: SN; Somalia: SO; Suriname: SR; South Sudan: SS; Sao Tome and Principe:
ST; El Salvador: SV; Sint Maarten (Dutch part): SX; Syrian Arab Republic: SY;
Eswatini: SZ; Turks and Caicos Islands: TC; Chad: TD; Togo: TG; Thailand: TH;
Tajikistan: TJ; Timor-Leste: TL; Turkmenistan: TM; Tunisia: TN; Tonga: TO; Türkiye:
TR; Trinidad and Tobago: TT; Tuvalu: TV; Taiwan, Province of China: TW; United
Republic of Tanzania: TZ; Ukraine: UA; Uganda: UG; United States of America: US;
Uruguay: UY; Uzbekistan: UZ; Holy See: VA; Saint Vincent and the Grenadines: VC;
Venezuela (Bolivian Republic of): VE; British Virgin Islands: VG; Viet Nam: VN;
Vanuatu: VU; World Intellectual Property Organization (WIPO) (International Bureau
of): WO; Samoa: WS; Nordic Patent Institute (NPI): XN; International Union for the
Protection of New Varieties of Plants (UPOV): XU; Visegrad Patent Institute (VPI):
XV; Unknown states, other entities or organizations: XX; Yemen: YE; South Africa:
ZA; Zambia: ZM; Zimbabwe: ZW; Version: V2_0"        }, {
          "type" : "string",
           "enum" : [ "AN", "CS", "DL", "DD", "DT", "RH", "SU", "YD", "YU" ],
           "description" : "Version: V2_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: V2_0”
        },
        "applicationNumberText" : {
          "type" : "string",
         "description" : "Description: Free format of application number; Version:
V2_0"
        }
      },
      "oneOf" : [ {
        "required" : [ "st13ApplicationNumber" ]
      }, {
        "required" : [ "applicationNumberText" ]
      } ]
    }
  }
}
```

22. 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" ],
  "$defs" : {
    "applicationNumber" : {
      "description" : "Description: Numbers used by IPOs in order to identify
each application received; Version: V2_0",
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType"

    }
  }
}
```

`applicationNumberType.json` (Example modular schema continued)

```
{
  "$id" : "applicationNumberType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "$defs" : {
    "applicationNumberType" : {
      "type" : "object",
      "unevaluatedProperties" : false,
      "description" : "Version: V2_0",
```

```
      "properties" : {
        "ipOfficeCode" : {
          "$ref" : "ipOfficeCode.json#/$defs/ipOfficeCode"
        }
      },
      "allOf" : [ {
        "oneOf" : [ {
          "allOf" : [ {
            "properties" : {
              "st13ApplicationNumber" : {
                "$ref" :
"st13ApplicationNumber.json#/$defs/st13ApplicationNumber"
              }
            },
            "required" : [ "st13ApplicationNumber" ]
          }, {
            "not" : {
              "required" : [ "applicationNumberText" ]
            }
          } ]
        }, {
          "allOf" : [ {
            "properties" : {
              "applicationNumberText" : {
                "$ref" :
"applicationNumberText.json#/$defs/applicationNumberText"
              }
            },
            "required" : [ "applicationNumberText" ]
          }, {
            "not" : {
              "required" : [ "st13ApplicationNumber" ]
            }
          } ]
        } ]
      } ]
    }
  }
}
```

`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" : {

      "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: V2_0",
"$ref" : "extendedWIPOST3CodeType.json#/$defs/extendedWIPOST3CodeType",

    }
  }
}
```

`extendedWIPOST3CodeType.json` (Example modular schema continued)

```
{
  "$id" : "extendedWIPOST3CodeType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "$defs" : {
    "extendedWIPOST3CodeType" : {
      "description" : "Version: V2_0",
      "anyOf" : [ {
        "$ref" : "wipoST3CodeType.json#/$defs/wipoST3CodeType"
      }, {
        "$ref" : "wipoFormerST3CodeType.json#/$defs/wipoFormerST3CodeType"
      }, {
        "$ref" : "igoCodeType.json#/$defs/igoCodeType"
      } ]
    }
  }
}
```

wipoST3CodeType.json (Example modular schema continued)

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

```
Taiwan, Province of China: TW; United Republic of Tanzania: TZ; Ukraine: UA;
Uganda: UG; United States of America: US; Uruguay: UY; Uzbekistan: UZ; Holy See:
VA; Saint Vincent and the Grenadines: VC; Venezuela (Bolivian Republic of): VE;
British Virgin Islands: VG; Viet Nam: VN; Vanuatu: VU; World Intellectual
Property Organization (WIPO) (International Bureau of): WO; Samoa: WS; Nordic
Patent Institute (NPI): XN; International Union for the Protection of New
Varieties of Plants (UPOV): XU; Visegrad Patent Institute (VPI): XV; Unknown
states, other entities or organizations: XX; Yemen: YE; South Africa: ZA; Zambia:
ZM; Zimbabwe: ZW; Version: V2_0",
      "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", "FM", "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", "MH", "MK",
"ML", "MM", "MN", "MO", "MP", "MR", "MS", "MT", "MU", "MV", "MW", "MX", "MY",
"MZ", "NA", "NE", "NG", "NI", "NL", "NO", "NP", "NR", "NU", "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",
  "type" : "object",
  "additionalProperties" : false,
  "$defs" : {
    "wipoFormerST3CodeType" : {
      "type" : "string",
      "description" : "Description: Netherlands Antilles: AN; Czechoslovakia:
CS; German Democratic Republic: DD; Federal Republic of Germany: DT; Southern
Rhodesia: RH; Soviet Union: SU; Democratic Yemen: YD; Yugoslavia/ Serbia and
Montenegro: YU; Version: V2_0",
      "enum" : [ "AN", "CS", "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: V2_0"
    }
  }
}
```

st13ApplicationNumberType.json (Example modular schema continued)

```
{
  "$id" : "st13ApplicationNumberType.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "$defs" : {
    "st13ApplicationNumberType" : {
      "type" : "string",      "description" : "Version: V2_0",

      "pattern" : "\\d{2}\\d{4}\\\\w{2}d{7}"
    }
  }
}
```

`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:
V2_0"
    }
  }
}
```

23. 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 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.

Documentation

24. 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.

25. A JSON Schema description 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 documentation using the "`description`" keyword.

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

Table 1. JSON Schema description items

Description item name	Value	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 or contact details of Organization to contact with questions about the schema	Optional
`SchemaReleaseNoteURL`	Location where schema release notes are published	Optional

[JSD-10] The description items such as date of schema created 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., as shown in the following example:

An example of description for `applicationBody` (Document Level Schema)

```
"description" : "Description: Body of a patent application; SchemaCreatedDate:
2012-07-13; SchemaLastModifiedDate: 2026-06-09; SchemaContactPoint:
standards@wipo.int; SchemaReleaseNoteURL:
https://www.wipo.int/standards/en/json_material/st97/v2-0/release-notes.html;
Version: V2_0 "
```

An example of description 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: V2_0"
```

An example of description for `appellateBodyCategoryType.json` (Enumeration Type definitions Schema)

```
"description" : " Office appeal board: Appeal board within the IP office; Court:
Court; Appeal Court: Second instance court; Supreme Court: Highest appellate court,
Version: V2_0;"
```

An example of description for Type definitions Schema

```
"description" : "Version: V2_0"
```

Schema file naming convention

26. The JSON Schema filename should use the lower camel case (LCC) convention.

27. 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.

28. A schema file may have version information which consists of either major and minor version indicators, or major, minor and patch version indicators. The version information applies only to document level schemas. 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. Test versions may be exceptionally provided only where there is a specific business need and it will not be considered an official version so that it will not be publicly available. A test version will be based on an official version but may comprise schema components of different official versions as a means of transitioning to an official version. Test versions MUST be denoted as such, in the Schema filename, putting the letter "T" with the revision number alongside the basis version 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_V2_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_V2_0_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".

JSON Schema Properties Structuring

29. 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" ],
  "$defs" : {
    "applicationNumber" : {
      "description" : "Description: Numbers used by IPOs in order to identify each
application received; Version: V2_0",
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType"
    }
  }
}
```

[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.

30. 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.

JSON SCHEMA CONSTRUCTS DESIGN RULES

Overview

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

Properties

32. 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.

Definitions

33. 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" : {

      "description" : "Description: Numbers used by IPOs in order to identify each
application received; Version: V2_0"
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",

    }
  }
}
```

34. 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.

35. 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.

Type Definitions

36. JSON Schemas may define reusable type definitions that are referenced from global property definitions. These reusable 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" ],
  "$defs" : {
    "applicationNumber" : {
      "description" : "Description: Numbers used by IPOs in order to identify each application received; Version: V2_0",
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType"
    }
  }
}
```

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

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".

37. 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".

Code Lists

38. 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 (_)}

Arrays

39. 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).

40. 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.

41. 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.

Objects

Property "wildcards"

42. 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.

43. Using the "`patternProperties`" keyword is not allowed, unless the schema is a mixed content type. 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 JSON Schema MUST NOT use the "`patternProperties`" keyword, unless the schema is a mixed content type.

Order of Properties

44. 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 JSON Schema SHOULD use the same order of properties as in the corresponding XML schema, if one exists.

JSON SCHEMA IDENTIFIERS

Overview

45. 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.

JSON INSTANCE DESIGN RULES

46. 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.

Order of Properties

47. 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.

JSON instance validation

48. 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.

[JIN-04] All modifications, updates, revisions, and new releases MUST first be submitted to the API 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.

REFERENCES

49. The following Standards and resources are of relevance to this Standard:

WIPO Standards

WIPO Standard ST.3	Two-letter codes for the representation of states, other entities and organizations
WIPO Standard ST.9	Bibliographic data on and relating to Patents and SPCs;
WIPO Standard ST.16	Identification of different kinds of patent documents;
WIPO Standard ST.26	Presentation of nucleotide and amino acid sequence listings using XML;
WIPO Standard ST.27	Exchange of Patent Legal Status Data;
WIPO Standard ST.60	Bibliographic data relating to marks;
WIPO Standard ST.61	Exchange of Trademark Legal Status Data;
WIPO Standard ST.67	Electronic management of the figurative elements of trademarks;
WIPO Standard ST.68	Electronic Management of sound marks;
WIPO Standard ST.69	Electronic management of motion and multimedia marks;
WIPO Standard ST.80	Bibliographic data relating to industrial designs;
WIPO Standard ST.87	Exchange of industrial design legal status data;
WIPO Standard ST.88	Electronic representation of industrial designs
WIPO Standard ST.90	Processing and communicating IP data using web APIs;
WIPO Standard ST.91	Digital Three-Dimensional (3D) models and 3D Images;
WIPO Standard ST.96	Processing of Intellectual Property information using XML

Standards and Industry Specifications

JSON Specification:	http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

JSON Schema Core, 2020-12 draft: https://json-schema.org/draft/2020-12/json-schema-core

JSON Schema Validation,2020-12 draft: https://json-schema.org/draft/2020-12/json-schema-validation

OpenAPI v3.2.0 Specification: https://spec.openapis.org/oas/v3.2.0.html

International Standard ISO/IEC 11179-5 “Information technology — Metadata registries (MDR) – Part 5: Naming and identification principles”;

International Standard ISO 3166-1 “Codes for the representation of names of countries and their subdivisions — Country Codes”;

International Standard ISO 639-1 “Codes for the representation of names of languages — Part 1: Alpha2-code”;
International Standard ISO 4217 “Codes for the representation of currencies and funds”;

International Standard ISO/IEC 10646 “Information technology — Universal Multiple-Octet Coded Character Set (UCS)”;

ISO/IEC 21778:2017 Information technology — The JSON data interchange syntax

ECMA-404 – The JSON Data Interchange Syntax

UBL-2.1 Universal Business Language — Universal Business Language Version 2.1

International Standard ISO/IEC 10646 “Information technology – Universal Multiple-Octet Coded Character Set (UCS)”;

Internet Engineering Task Force (IETF) “Request For Comments (RFC) 2119 — Key words for use in RFCs to Indicate Requirement Levels”

[Annex I follows]