C

WIPO
世界
知识产权
组织

CWS/7/5
原文：英文
日期：2019年5月20日

世界知识产权组织标准委员会（CWS）

第七届会议
2019年7月1日至5日，日内瓦

关于JSON规范的提案

国际局编拟的文件

导　言

1.　国际局观察到越来越多的知识产权局已经开始使用JavaScript对象简谱（JSON）格式用于数据传播，特别是通过网络服务。JSON是基于文本的数据交换格式，它被认为是轻量级格式，尤其是与XML格式相比。但是，目前还没有广泛接受的JSON架构行业标准。

2.　XML4IP工作队自2013年起，已有考虑除工作队管理的标准XML架构之外也使用JSON格式。工作队注意到JSON是在申请和网络服务之间交换数据时的首选选项，而XML则是各知识产权局之间分享和储存文献时的首选格式。

3.　产权组织标准委员会（CWS）在创建第56号任务执行以下任务时，也考虑了将JSON格式用于网络服务：

> “为支持机器对机器通讯的数据交换编写建议，重点是：
>
> – 采用JSON和/或XML的消息格式、数据结构和数据字典；以及
>
> – 资源的统一资源标识符（URI）命名约定。”

4.　考虑到各知识产权局对JSON格式不断增加的需求和用途，以及上文第56号任务的说明，XML4IP工作队编写了一份关于使用JSON的知识产权数据的新产权组织标准的工作文件草案，作为本文件附件提供。该规范草案以美国专利商标局（美国专商局）所提提案为基础编制，供标准委员会第七届会议审议并提出评论意见。

新 JSON 标准工作草案

5. 构成工作草案的一套准则是与产权组织标准 ST.96 紧密契合的，包括与 ST.96 附件一“XML 设计规则和约定”中定义非常相似的命名约定。XML4IP 工作队强调了 JSON 对象和 XML 实例之间兼容性的重要性，并建议重复利用 ST.96 的架构组件名称，但以小驼峰式命名法格式表明其是 JSON 对象的除外。

6. 产权组织标准 ST.96 包括主体和六个附件。由于该规范的内容与 ST.96 附件一类似，JSON 标准的最终组成很可能会包含更多建议，例如像 ST.96 附件五之类的实施规则和准则。

标准的范围

7. XML4IP 工作队认为，该标准应该为各知识产权局和创建或存储使用 JSON 资源的知识产权数据的其他利益方提供指导。

标准的目标

8. 工作队同意该标准草案应当旨在提供起草 JSON 架构的指导方针，以便简化各知识产权局制作 JSON 架构的工作。尽管该工作草案尚在编制的初级阶段，但它致力于实现用于 XML 和 JSON 两种格式的单一词汇表，以避免任何命名中的混淆。工作队寻求扩展该规范以包含 JSON 架构，最好是从原子级构建。

9. 更具体而言，该标准的宗旨是：

- 为 JSON 数据标记标准化提供指导；
- 通过制定 JSON 设计原则确保一致性；
- 通过促进在各知识产权局之间以及公开提供的数据中重复利用 JSON 资源，提高数据交换效率；
- 借助重复利用不断促进易用性和对数据的理解。

标准的结构

10. 最新工作草案（草案 0.1）中包含以下五个章节以及一项导言：

- 基于产权组织标准 ST.96 的命名约定；
- JSON 架构设计规则：用于架构本身；
- JSON 架构构造设计规则：用于对象和类型；
- JSON 架构标识符：定义用于定位 JSON 资源的统一资源标识符（URI）；以及
- JSON 实例设计规则：JSON 实例的结构和限制。

11. 除此之外，还有三份附录：

- 附录 A：提供 XML 架构和 JSON 架构之间映射的一系列表格；
- 附录 B：所用的代表术语，即大类及其数据类型；
- 附录 C：取代术语全称时必须使用的标准首字母缩写和缩略语列表。

进一步讨论和编制工作

12. 国际局于 2019 年 2 月组织了一次 XML4IP 工作队在线会议讨论美国专利商标局编拟的提案。XML4IP 工作队成员在 2019 年 3 月于首尔举行的会议上审议了美国专利商标局提交的初始提案。工作队同意基于产权组织标准 ST.96 编制 JSON 架构，并同意从简单结构的 XML 组件，即原子级组件入手。

13. 产权组织标准 ST.96 中定义了约 1,800 个 XML 架构组件。作为 XML4IP 任务牵头人，国际局将与 XML4IP 工作队成员协作，研究促进产权组织标准 ST.96 XML 架构向 JSON 架构转化的解决方案。

14. XML4IP 工作队将继续改进规范草案，并计划提交最终提案供标准委员会第八届会议审议。

15. 请标准委员会：

(a) 注意本文件的内容；

(b) 就 JSON 规范的工作草案提出评论意见；

(c) 鼓励其成员参与关于 JSON 规范的讨论，检验 JSON 架构；并向 XML4IP 工作队提供反馈意见；

(d) 要求 XML4IP 工作队提交最终提案供标准委员会第八届会议审议。

[后接附件]

WIPO STANDARD ST.XX

PROCESSING OF INTELLECTUAL PROPERTY INFORMATION USING JSON

JSON DESIGN RULES AND CONVENTIONS

Working Draft – version 0.1

Editorial Note – prepared by the International Bureau

This Working Draft is prepared by the International Bureau and shared for information at the seventh session of the CWS only in English. This Draft will be further updated in due course and the final draft will be submitted for consideration by the CWS at its eighth session.

TABLE OF CONTENTS

1 INTRODUCTION

This draft 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 considering the rules and conventions set out in WIPO Standard ST.96 – “Recommendation for the processing of Industrial Property Information using XML (Extensible Markup Language)”.

This proposed purpose for this standard is to:

- Provide guidance on JSON data mark-up standardization;
- Ensure consistency by establishing design principles for JSON;
- 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 terminology is used:

- the term “JSON schema” refers to any language for describing the structure and constraining the contents of JSON documents and the JSON schema Core, version draft-07, available at http://json-schema.org/latest/json-schema-core.html. JSON schema version is subject to change because it has not achieved RFC status.; it has not been adopted by an IETF Working Group. It should also noted that earlier drafts of JSON schema may not be completely compatible.
- the term “schema” (alone) refers specifically to a JSON schema.
- atomic property names are the lowest level of granularity within the JSON Objects.
- In this document, "SHALL", "SHALL 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 document:

- <>: Indicates a placeholder descriptive term that, in implementation, will be replaced with a specific instance value.
- “ ”: Indicates that the text included in quotes must be used verbatim in implementation.
- { }: Indicates that the items are optional in implementation.
- Consolas font: Indicates JSON keywords, JSON property names.

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 schema design rules

  (c) JCD for construct design rules

  (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 aims to provide guidance to Intellectual Property Offices and other Organization that create or modify Intellectual Property data as JSON resources. Compliance with this standard is required for data exchange between IPOs using JSON.

This draft standard excludes the following:

(a) architectural concerns;
(b) implementation languages; and
(c) tools for producing JSON schema.

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 facet of JSON technology. The general rules and guidelines, listed below, provide the common foundation for JSON document and JSON data structure development for all data to include intellectual property (IP) data and non-IP data.

5.2 JSON Naming Conventions

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

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 JSON properties consist of the following terms:

- *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.
- *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/Attribute Term* identifies characteristics of the Object Class.
- *Representation Term* categorizes the format of the data element into broad types. Representation Terms listed in Appendix B to this document should be used for JSON Design Rules and Conventions Standard.

[JGD-01] Property names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary.

[JGD-02] Property names SHOULD consist only of nouns, adjectives, and verbs in the present tense.

[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 a property name SHOULD be no more than 35 characters.

[JGD-05] Property names MUST NOT contain consecutive redundant words.

[JGD-06] Property names SHOULD be concise and self-explanatory.

[JGD-07] Property names MUST use the lowerCamelCase (LCC) convention. For example, "currencyCode":"EUR".

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

[JGD-09] Acronyms and abbreviations at the beginning of a property name MUST appear in all lower case. All other acronym and abbreviation usage in a property name MUST appear in upper case in Appendix C.

[JGD-10] 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-11] A Qualifier Term MAY be attached to an Object Class Term or a Property Term if necessary to make a name unique.

[JGD-12] 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.

[JGD-13} A Qualifier Term SHOULD precede the associated Object Class Term or Property Term.

Example Listing
Employee Last Name – employeeLast**Name** Claims Total Quantity - claimTotal**Quantity** Application Patent Case Filing Date - applicationPatentCaseFiling**Date** Fiscal Year Budget Period Total Amount – fiscalYearBudgetPeriodTotal**Amount** Note: the representational term is highlighted with **bold** font.

[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] The Representation Terms in Appendix B MUST be used for atomic property names.

[JGD-16] Word(s) in a property name SHOULD be in singular form unless the concept itself is plural. For example: goodsServices, totalMarkSeries.

[JGD-17] The name of a property which contains an array or a collection of contextually related components SHOULD have the “plurality” representation in its name using “Bag”. For example, EmailAddressBag represents a collection of EmailAddress properties

[JGD-18] Connecting words like “and”, “of” and “the” SHOULD NOT be used in property names unless they are part of the business terminology.

[JGD-19] Property names MUST NOT be translated, changed or replaced for any purpose.

[JGD-20] Property names MUST NOT refer to article and rule numbers. For example, PCT Article 34.

[JGD-21] Collection resources that include collection-wide metadata (e.g. total number of items) should be wrapped within a JSON object envelope. Custom HTTP headers MUST NOT be used for conveying this metadata.

[JGD-22] Levels of nesting SHOULD be kept to a minimum. Prefer inline values over having a values that are single-property objects. For example, instead of "inventor": { "fullName": "Thomas Edison" }, "inventorFullName": "Thomas Edison" is preferred.

[JGD-23] Names within the URL SHOULD be used to provide context for the data represented by JSON. Avoid repeating terms within JSON that are implicitly understood based on the context.

For example, consider a resource http://patent-classification.uspto.gov/classifications/cpc with the following representation:

```
{
    "cpcClassificationBag" : [
    {
                        "cpcClassification": {
                                "class" : …
                                "subclass": …
                        }
                },
            {
                        "cpcClassification": {
                        "class": …
                        "subclass": …
                        }
                },
    ]
}
```

Given that the URL makes it clear that the resource is a collection of classifications, specifically CPC classifications, the property names are providing redundant information in this context. Thus, the JSON representation could be simplified to:

```
[
                {
                  "class": …
              "subclass": …
    },
    {
                  "class": …
              "subclass": …
                }
```

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. Business users rely on IPO systems for well-defined JSON schema for data structure validation and common vocabulary across systems. Every JSON schema is versioned to reduce the impact to systems as a result of structural changes to an existing schema.

JSON schema documents should comply with the industry schema standard, JSON schema. The latest version available at time of publishing this document is draft-0.7 and this draft standard refers to this version.

[JSD-01] JSON schema documents MUST conform to JSON schema specifications: JSON schema Core, version draft-07, available at http://json-schema.org/latest/json-schema-core.html, and JSON schema Validation, version draft-07, available at http://json-schema.org/latest/json-schema-validation.html.

[JSD-02] The schema document MUST indicate that they conform to version draft-07 of JSON schema using the $schema keyword.

Example: Indicating the version of JSON schema
`"$schema": "http://json-schema.org/draft-07/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:

inventor_V1_0.json

```
{
    "$id": "urn:us:gov:doc:uspto:patent",
     "title": "inventor_V1_0",
   "$schema": "http://json-schema.org/draft-07/schema#",
    "type": "object",
    "required": [ "inventor" ],
    "properties": {
        "inventor":  {
          "description": "The individual or entity responsible for creating the
matter that protection is sought for.",
            "properties": {
                "personName": {
              "properties": {
                 "prefixName": { "type": "string" },
                 "firstName": { "type": "string" },
                 "middleName": { "type": "string" },
                 "lastName": { "type": "string" },
                 "suffixName": { "type": "string" }
              },
              "required": [ "firstName","middleName","lastName" ],
              "additionalProperties": false
            },
                "address": {
              "properties": {
                 "physicalAddress": {
                    "properties": {
                       "addressLineText": { "type": "string" },
                       "cityName": { "type": "string" },
                       "geographicRegion": { "type": "string" },
                       "postalCode": { "type": "string" },
                       "countryCode": { "type": "string" }
                    },
                    "required": [ "addressLineText", "cityName",
"geographicRegion", "postalCode", "countryCode"],
                    "additionalProperties": false
                 },
                 "telecommunicationAddress": {
                    "properties": {
                       "internationalDialingCode": { "type": "string" },
                       "areaCode": { "type": "integer" },
                       "exchangeNumber": { "type": "integer" },
                       "extensionNumber": { "type": "integer" },
                       "telecommunicationServiceCategory": { "type": "string" }
                    },
                    "required": [ "internationalDialingCode",
"areaCode","exchangeNumber","extensionNumber","telecommunicationServiceCategory"]
,
                    "additionalProperties": false
                 }
              },
              "required": [ "physicalAddress", "telecommunicationAddress" ],
              "additionalProperties": false
            }
            },
            "required": [ "personName", "address" ],
            "additionalProperties": false
        }
    }
}
```

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 Inventor schema below. This JSON schema is built upon smaller JSON schema modules.

inventor_V2_0.json

```
{
"$id": "urn:us:gov:doc:uspto:patent",
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "inventor_V2_0",
"description": "Schema Name: Patent Application; Schema Published on: June 30,
2010;  Version Number: 1.0; Description: Developed by: USPTO; Point of
Contact:",
"definitions":  {
     "inventor":  {
          "$id": "#inventor",
          "description": "The individual or entity responsible for creating the
matter that protection is sought for.",
          "$ref": "#/definitions/inventorType"
          },
     "inventorType":  {
          "type": "object",
          "properties":  {
               "personName":  {"$ref":
"urn:us:gov:doc:uspto:common#personName"},
               "organizationName":  {"$ref": "urn:us:gov:doc:uspto:
common#organizationName"},
               "designation":  {"$ref":
"urn:us:gov:doc:uspto:patent#designation"},
               "stateDesignation":  {"$ref":
"urn:us:gov:doc:uspto:patent#stateDesignation"},
               "registrationNumber":  {"$ref":
"urn:us:gov:doc:uspto:patent#registrationNumber"},
               "sequence":  {"type": "string"}
               },
          "additionalProperties": false,
          "required":
["personName","designation","stateDesignation","registrationNumber","sequence"]
          }
     },
"type": "object",
"minProperties": 1,
"additionalProperties": false,
"properties":  {
     "inventor":  {"$ref": "#/definitions/inventor"}
     }
}
```

personName_V1_0.json

```
{
    "$id": "urn:us:gov:doc:uspto:common",
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "personName_V1_0",
   "description": "...",
"definitions":  {
     "personName":  {
          "$id": "#personName",
          "description": "...",
          "$ref": "#/definitions/personNameType"
          },
     "personNameType":  {
          "type": "object",
          "properties":  {
                "prefixName": { "$ref":
"urn:us:gov:doc:uspto:common#prefixName" },
                "firstName": { "$ref":
"urn:us:gov:doc:uspto:common#firstName" },
                "middleName": { "$ref":
"urn:us:gov:doc:uspto:common#middleName" },
                "lastName": { "$ref": "urn:us:gov:doc:uspto:common#lastName" },
                "suffixName": { "$ref":
"urn:us:gov:doc:uspto:common#suffixName" }
             },
          "additionalProperties": false,
          "required": ["firstName","lastName"]
          }
     },
"type": "object",
"minProperties": 1,
"additionalProperties": false,
"properties":  {
     "personName":  {"$ref": "#/definitions/personName"}
     }
}
```

JSON schemas should use the "definitions" 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 "definitions" keyword that includes a reusable definition for each property and property type.

[JSD-05] Developers MUST use existing resources, wherever applicable, prior to creating new JSON schemas.

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

6.3 Documentation

JSON schema documents 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 will 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 MUST include JSON schema header documentation using the "description" keyword.

Table 1 lists the items that should be included in the header section of all schemas.

Table 1. JSON schema header documentation items

Header item name	Description	Required/Optional
Schema Name	The schema file	Optional
Published on	Publication date of schema	Required
Version Number	Major and Minor version number of the schema	Required
Description	Plain text description of the information described by the schema	Required
Developed by	Name of the organization or office that developed the schema	Optional
Point of Contact	Name of Organization to contact with questions about the schema	Optional

The header items (schema name, published on, etc.) below 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
`"description" : "Schema Name: Patent Application Schema; Published on: June 30, 2010; Version Number: 1_1; Description: Patent Applications; Developed by: USPTO; Point of Contact:"`

6.4 File Name

File names must be compatible with multiple operating systems, applications and development platforms. Therefore filenames must be restricted to ASCII characters most commonly used for filenames. Lower-case file names are required to reduce complications across various file systems.

[JSD-09] The characters used in file names MUST be restricted to the following set: {a-z, A-Z, 0-9, underscores (_) and period (.)}.

There are two allowed approaches for JSON schema definition. JSON schemas may contain the definition of properties based on root property name (logical grouping name), for which the filename would be the name of the root property name followed by “DataElements”, version number and schema extension e.g. addressDataElements_V1_0.schema.json. The other scenario for schema would be the definition of a subschema. The keyword associated with the subschema should be defined in the same schema file. The filename should reflect the keyword, followed by version number and schema extension. e.g. trademarkApplication_V1_0.schema.json.

[JSD-10] The filename SHOULD be the combination of the root property name, version number and JSON extension (e.g., patentApplication_V1_0.schema.json). Use of system and project names is prohibited, because the system and project name may change over the life of the system.

6.5 JSON schema versioning

JSON schemas should reflect the ongoing changes to the business requirements. Versioning plays an important role in accommodating the change and growth inherent in business requirements. Changes to the schema construct can be better tracked by proper versioning of JSON schemas. JSON schema version should be denoted by both a major and a minor version number in the format of ‘M_m’. ‘M’ represents the major version number of the schema in digits and starts from 1, whereas ‘m’ represents the minor version number in digits and starts from 0. JSON schema version numbers should be part of the file name.

[JSD-11] JSON schemas MUST include major and minor version numbers separated by ‘_’ in their file name. The version number preceded by an underscore ‘_V<major version>_<minor version>’ is appended to the file name, e.g. trademarkApplication_V1_0.json.

Version number is to be included in the header documentation of the schema.

[JSD-12] Version number included in the title schema header documentation MUST include major and minor version numbers separated by '_'.

[JSD-13] The schema file name and header documentation MUST contain matching version information.

The original version of any schema should be denoted with a version number of <1_0>. Any subsequent change to the schema should be represented by incrementing either the minor version number or the major version number depending on the nature of the change.

[JSD-14] New minor versions of a schema MUST only add new optional properties to prior versions, or make some cardinalities or value constraints less restrictive.

[JSD-15] New minor versions of schema MUST be able to validate instance documents created with preceding minor versions of that schema with the same major version i.e. minor versions should be backwards compatible with the previous major version. However, instance documents should not be expected to validate against versions of a schema preceding the one they were created with.

When there is a major change in the schema, the major version number should be incremented while resetting the minor version number to 0. For example, a schema may go through the following version changes: <1_0>; <1_1>; <1_2>; <1_3> <2_0>; <2_1>; <2_2>; <3_0>; <3_1> … etc. where the version 2.0 and 3.0 are major versions.

[JSD-16] A major version of a schema MUST be incremented if new mandatory properties are added, any properties are removed or cardinalities or value constraints are changed to be more restrictive.

[JSD-17] When creating a new schema, one SHOULD include the most recent versions of all the other referenced schemas.

6.6 JSON Schema Document Properties Structuring

JSON schemas should have property "type" : "object" to ensure that JSON is used only for complex structures, not individual values. Taken from **personName_V1_0.json** example above:

```
"type": "object",
"minProperties": 1,
"additionalProperties": false,
"properties":  {
     "personName":  {"$ref": "#/definitions/personName"}
     }
}
```

[JSD-18] The outermost schema object must have a "type" keyword whose value is "object".

JSON schemas should have property "minProperties" : 1 to ensure that an empty value is not valid according to the schema.

[JSD-19] The outermost schema object must have a "minProperties" keyword whose value is 1.

JSON schema extensions (customizations) should not be used. All keywords used must be defined by the JSON schema specification.

[JSD-20] JSON schema extensions (customizations) MUST not be used.

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] JSON schemas SHOULD use existing properties and schemas to the maximum extent possible.

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

In some cases, where the object does not belong to a logical grouping and needs to be reused across the business, that property can be declared in its own schema file.

[JSC-03] A property that does not fall into a logical group MAY be declared in its own schema file.

Each property should have a "global" definition that is defined within the "definitions" array. This will allow the property name to be reused in many parents and have a consistent definition across all of them. Please see the "inventor" property in the example below.

[JSC-04] Each property listed in a properties keyword SHOULD refer to a global definition that is defined within the "definitions" keyword. That global definition SHOULD have the same name as the property.

An example of a property referring to a global definition

```
{
"$id": "urn:us:gov:doc:uspto:patent",
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "inventor_V1_0",
"description": "...",
"definitions":  {
     "inventor":  {
          "$id": "#inventor",
          "description": "...",
          "$ref": "#/definitions/inventorType"
          },  ...
     },
"type": "object",
"minProperties": 1,
"additionalProperties": false,
"properties":  {
     "inventor":  {"$ref": "#/definitions/inventor"}
     }
}
```

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

[JSC-05] The global definition of a property SHOULD consist of the "id" and "description" of the property.

Properties must have types. They can either be defined directly within the definition of the property, or be handled through a reference to a global property definition.

[JSC-06] 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.

7.3 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-07] A schema MAY define global type definitions in order to reuse content models across many properties.

A reusable type definition

```
"definitions":  {
     "inventorType":  {
          "type": "object",
          "properties":  {
               "personName":  {"$ref": "urn:us:gov:doc:uspto:common#personName"},
               "organizationName":  {"$ref":
"urn:us:gov:doc:uspto:common#organizationName"},
               "designation":  {"$ref": "urn:us:gov:doc:uspto:patent#designation"},
               "stateDesignation":  {"$ref":
"urn:us:gov:doc:uspto:patent#stateDesignation"},
               "registrationNumber":  {"$ref":
"urn:us:gov:doc:uspto:patent#registrationNumber"},
               "sequence":  {"type": "string"}
               },
          "additionalProperties": false,
          "required": ["personName","designation", "stateDesignation",
"registrationNumber", "sequence"]
          }
     },
```

[JSC-08] Definitions that represent types MUST have names that are in LCC convention + Suffix "Type".

7.4 JSON Primitive Type

[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".

7.5 Code Lists

In certain cases, it is advantageous to restrict a value to an enumerated list 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 W3C or OASIS should be used.

[JSC-11] WIPO Standard ST.3 MUST be used for representing IPOs, states, other entities, organizations and for priority and designated country/organization.

[JSC-12] ISO 3166-1-Alpha-2 Code Elements (2 letter country codes) MUST be used for the representation of the names of countries for addressing and citizenship.

[JSC-13] ISO 639-1 (2-Letter Language Codes) MUST be used for Language Codes.

[JSC-14] ISO 4217-Alpha (3-Letter Currency Codes) MUST be used for Currency Codes.

[JSC-15] Documentation SHOULD NOT be substituted for code lists using enumeration.

Below is an example where the documentation is used instead of code lists, using enumeration for language options which is NOT recommended.

Bad example: Code list is documented not defined

```
"description": "Possible values for language codes: en – English, es –  Esperanto",
```

Enumeration used for a code list

```
"description": "ISO 639-1:2002 Part 1: Alpha-2 Language Codes",
"enum": [ "en", "es"]
```

[JSC-16] The JSON `enum` keyword SHOULD be used for defining the code lists.

[JSC-17] 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 (_)}

7.6 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-18] 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-19] 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-20] The "additionalItems" keyword SHOULD NOT be used when “items” is a single schema object.

7.7 Objects

7.7.1 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-21] A JSON schema SHOULD 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.

[JSC-22] A schema MUST NOT use the "patternProperties" keyword.

7.7.2 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-23] 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.

Once an ID is established and associated with JSON schemas, it must remain the same.

[JID-02] Published IDs MUST never be changed.

JSON constructs belonging to a specific schema ID should have unique names.

[JID-03] Within a schema, all names MUST be unique.

8.2 ID versioning

Many-to-one relationships exist between schemas and IDs. Distinct schemas may be associated with a single ID. Within the ID, each schema may go through several major and minor version changes. It will be difficult to associate all the version changes of the schemas to the versioning of the ID. There will never be two schemas with the same file name and the same ID. Name conflicts should not be an issue across all the namespaces as each schema will clearly be identified by its name along with the version number and its ID.

[JID-14] There SHOULD NOT be any versioning of the IDs.

9 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.

[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 document.

10 REFERENCES

- JSON Specification: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
- JSON schema Core, v.7 draft: http://json-schema.org/latest/json-schema-core.html
- JSON schema Validation, v.7 draft: http://json-schema.org/latest/json-schema-validation.html
- WIPO ST.96: http://www.wipo.int/standards/en/st96

11 APPENDICES

11.1 Appendix A – Mapping XML schema to JSON schema

This section describes the mapping between XML schemas and JSON schemas and explains how to convert from one to the other.

Table 2. Mapping of XML and JSON schema Concepts

XML Schema	JSON Schema
Schema	Schema
Namespace	ID
Element	Property
Attribute	Property
Global Element Declaration	Definition of the property under the "definitions" keyword.
Global/Named Type Definition	Definition of the type under the "definitions" keyword.
Repeating Element	Property of type "array"

XML Schema	JSON Schema
Complex Content Element	Property of type "object" (or array of objects)
Simple Content Element	Property of type "string", "number", "integer" or "boolean"

Table 3. Mapping of XML and JSON schema Keywords

XML Schema	JSON Schema
xs:annotation	description
@id	id
xs:complexType / xs:sequence or xs:all	properties
xs:complexType / xs:choice	Properties/oneOf or anyOf
@minOccurs (> 0), @use	required
@minOccurs, @maxOccurs	minItems, maxItems
@ref	$ref
@type	type
xs:minLength	minLength
xs:maxLength	maxLength
xs:length	minLength + maxLength
xs:pattern	pattern
xs:enumeration	enum
xs:minExclusive	minimum, exclusiveMinimum="true"
xs:maxExclusive	maximum, exclusiveMaximum="true"
xs:minInclusive	minimum
xs:maxInclusive	maximum

Table 4. Mapping of XML and JSON schema Simple Types

XML Schema	JSON Schema
xs:integer	integer
xs:decimal, xs:float, xs:double	number
xs:boolean	boolean
xs:date, xs:dateTime, xs:time	string, with format "date-time"
xs:anyURI	string, with format "uri"
all others	string

11.2 Appendix B – 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
Directory	Always preceded by PATH	String
Document	A CLOB stands for "character large object," which is a specific data type for almost all databases. Quite simply, a CLOB is a pointer to text stored outside of the table in a dedicated block. Used for XML documents. Comprised of textual information of International Trademark Registration being exchanged. XML tags identify the data items concerned with such information. TIS - Madrid development team may define the attribute XML_DOC as CLOB, pointer to Tagged Data stored outside of the table in a dedicated block.	String
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. Recommended values are Y, N, and, “?” if needed.	boolean
Measure	A measure is a numeric value determined by measuring an object along with the specified unit of measure. MeasureType is used to represent a kind of physical dimension such as temperature, length, speed, width, weight, volume, latitude of an object. More precisely, MeasureType should be used to measure intrinsic or physical properties of an object seen as a whole.	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.	Number, String
Percent	A number which represents a part of a whole, which will be divided by 100.	Number

Term	Definition	Data Type
Quantity	A quantity is a counted number of non-monetary units, possibly including fractions. Quantity is used to represent a counted number of things. Quantity should be used for simple properties of an object seen as a composite or collection or container to quantify or count its components. Quantity should always express a counted number of things, and the property will be such as total, shipped, loaded, stored. **QuantityType** should be used for components that require unit information; and **xsd:nonNegativeInteger** should be used for countable components which do not need unit information.	Number
Rate	A quantity or amount measured in relation to another quantity or amount.	Number
Text	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.	Date
DateTime	The captured date and time of an event when it occurs.	Date
URI	The Uniform Resource Identifier that identifies where the file is located.	String

11.3 Appendix C – List of Acronyms and Abbreviations

Short Form	Long Form
Alt	Alternate Text for image
B	Bold
BioDeposit	Biological Deposit
Br	Break
CDX	CambridgeSoft proprietary ChemDraw file format
CPC	Cooperative Patent Classification
DD	Definition Description
Del	Deleted text
DL	Definition List
DOI	Digital Object Identifier
DT	Definition Term
DTD	Document Type Definition
DWF	Design Web Format
DWG	Drawing
ECLA	European Classification
ExtRef	References that are external to the current document
H<n>	The “n” indicates the level of Heading with a specific value from 1 to 15 digit number. It means, in the enumeration value, this abbreviation represents one of H1 to H15. For example “H1” means “Heading 1”.
I	Italic
ID	Identifier for system identification
IDREF	Identifier Reference
IDREFS	Identifier References

Short Form	Long Form
IGES	Initial Graphic Exchange Specification
Ins	Inserted text
IP	Industrial Property
IPC	International Patent Classification
IPCR	International Patent Classification Reform
IPO	Industrial Property Office
IPR	Industrial Property Right
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LCC	Lower Camel Case
LI	List Item
LOR	License Of Right
MPEG	Moving Picture Experts Group
MOL	File format for holding information about the atoms, bonds, connectivity and coordinates of a molecule
NB	File format for Mathematica notebooks
NPL	Non Patent Literature
O	Over score
OASIS	Organization for the Advancement of Structured Information Standards
OCR	Optical character recognition
OL	Ordered List
P	Paragraph
PAN	Primary Account Number
PCT	Patent Cooperation Treaty
PKCS7	In cryptography, PKCS is a group of public-key cryptography standards and PKCS #7 (PKCS7) is for the Cryptographic Message Syntax Standard which describes general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes.
Pre	Preformatted text
S	Strike through text
SEQL	Sequence listing
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
U	Underlined
UCC	Upper Camel Case
UL	Unordered List

Short Form	Long Form
UPOV	The International Union for the Protection of New Varieties of Plants
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WIPO	World Intellectual Property Organization
WMV	Windows Media Video
XML	eXtensible Markup Language

[附件和文件完]