CWS/10/6 REV.
ORIGINAL : ANGLAIS
DATE : 10 NOVEMBRE 2022

Comité des normes de l'OMPI (CWS)

Dixième session
Genève, 21 – 25 novembre 2022

PROPOSITION DE NOUVELLE NORME DE L'OMPI CONCERNANT LE FORMAT JSON

Document établi par le Bureau international

RÉSUMÉ

1. Au nom de l'Équipe d'experts chargée de la norme XML4IP, le Bureau international présente une proposition finale relative à la nouvelle norme de l'OMPI concernant le format JavaScript Object Notation (JSON), pour examen à la dixième session du Comité des normes de l'OMPI (CWS). La proposition comprend un ensemble de règles de conception et une série de schémas JSON, qui sont fondés sur les schémas XML de la norme ST.96 de l'OMPI, ainsi que des exemples d'instances JSON.

CONTEXTE

2. À sa cinquième session tenue en 2017, le CWS a approuvé l'ajout de la nouvelle tâche n° 56 en vue de définir le cadre dans lequel l'Équipe d'experts chargée de la norme XML4IP élaborerait une nouvelle norme de l'OMPI qui offre un ensemble de recommandations relatives au traitement et à la communication des données de propriété intellectuelle aux API Web (interfaces de programmation d'application). La description de cette tâche indique que le format JSON peut être utilisé en tant que charge utile.

3. À sa septième session, le CWS a créé la nouvelle tâche n° 64, dont la description est reproduite ci-après, et a confié la tâche à l'Équipe d'experts chargée de la norme XML4IP (voir les paragraphes 58 et 59 du document CWS/7/29) :

> "*Préparer une proposition de recommandations concernant les ressources JavaScript Object Notation (JSON) compatibles avec la norme ST.96 de l'OMPI pour le dépôt, le traitement, la publication et/ou l'échange d'informations sur la propriété intellectuelle.*"

4. À sa huitième session tenue en 2020, le CWS a adopté la nouvelle norme ST.90 de l'OMPI intitulée "Recommandations relatives au traitement et à la communication des données de propriété intellectuelle aux API Web (interfaces de programmation d'application)". La norme ST.90 de l'OMPI comprend des exemples aux formats XML et JSON. Si la norme ST.90 de l'OMPI se réfère à la norme ST.96 de l'OMPI en ce qui concerne les schémas XML, il n'existe pas de norme de référence en ce qui concerne le format JSON, puisque aucune norme de l'OMPI en ce sens n'existait alors. La norme ST.90 de l'OMPI comprend une note de bas de page indiquant : "*La norme JSON de l'OMPI est en cours d'examen, mais elle sera basée sur la norme ST.96 de l'OMPI*."

5. À sa neuvième session, tenue en 2021, le CWS a noté qu'une proposition finale pour la nouvelle norme JSON serait présentée pour examen et adoption à sa dixième session (voir le paragraphe 20 du document CWS/9/25). Depuis la neuvième session du CWS, dans le cadre de la tâche n° 64, l'Équipe d'experts chargée de la norme XML4IP a été en mesure de mettre au point la proposition concernant la norme au terme d'une série de délibérations sur ce thème, qui ont eu lieu à l'occasion de réunions, dans le cadre d'échanges de courriers électroniques et sur le Wiki.

PROPOSITION CONCERNANT LA NOUVELLE NORME DE L'OMPI

6. Le format JSON a été progressivement adopté et utilisé par les offices de propriété intellectuelle et le secteur de la propriété intellectuelle alors que le format XML (eXtensible Markup Language), fondé sur les normes XML de l'OMPI, est encore largement utilisé. La norme ST.96 de l'OMPI offre une recommandation en ce qui concerne l'utilisation des ressources XML pour le dépôt, la publication, le traitement et l'échange d'informations concernant différentes catégories de titres de propriété intellectuelle, c'est-à-dire les brevets, les marques, les dessins et modèles industriels, les indications géographiques et le droit d'auteur. Les offices de propriété intellectuelle appliquent la norme ST.96 de l'OMPI telle qu'elle est publiée ou en y apportant certaines adaptations, le cas échéant.

7. L'Équipe d'experts chargée de la norme XML4IP a élaboré la proposition de norme JSON en tenant compte de la nécessité d'assurer la conformité et la compatibilité des données entre les formats XML et JSON, de façon à favoriser les échanges de données entre les offices de propriété intellectuelle et la diffusion des données par les offices de propriété intellectuelle dans ces deux formats. La conformité et la compatibilité des données peut être assurée au moyen de schémas XML et JSON compatibles qui seront utilisés pour valider les instances XML et JSON.

8. Au moment de l'élaboration du présent document, il n'existe aucune norme JSON au niveau international qui soit approuvée par le secteur. Seuls des projets de spécifications existent, tandis que le format JSON évolue constamment. Le projet 2020-12 est la dernière version de la proposition de spécification relative au schéma JSON et la dernière version de la norme ST.96 de l'OMPI est la version 5.0. C'est pourquoi la proposition concernant la norme JSON s'appuie sur la proposition de spécification relative au schéma JSON et sur la version 5.0 de la norme ST.96. Il convient de noter que la version 6.0 de la norme ST.96 sera publiée en octobre 2022.

Objectifs

9. La norme faisant l'objet de la proposition vise à formuler une série de recommandations concernant la présentation des données de propriété intellectuelle au format JSON. Elle a pour principal objectif ~~d'apporter les avantages suivants~~ :

- de donner des orientations concernant la conception et l'élaboration de pratiques recommandées relatives aux données de propriété intellectuelle au format JSON~~de donner des orientations concernant le balisage des données au format JSON~~;

- d'assurer la conformité au moyen de schémas et d'instances au format JSON fondés sur la norme ST.96 en ce qui concerne l'échange des données de propriété intellectuelle;
- de recommander des principes de conception relatifs à l'extension des schémas JSON élaborés ou à la création de nouveaux schémas JSON conformes; et
- de rationaliser l'échange de données en favorisant la réutilisation des ressources JSON entre les offices de propriété intellectuelle, ainsi que des données communiquées au public.

Portée

10. Cette nouvelle norme devrait apporter des orientations aux offices de propriété intellectuelle et aux organisations concernées qui créent ou modifient des données de propriété intellectuelle en tant que ressources JSON. Il est nécessaire de suivre cette norme pour assurer l'échange de données entre les offices de propriété intellectuelle qui utilisent des ressources dans le format JSON en tant que schémas, instances, messages ou charges utiles pour les API.

11. La norme faisant l'objet de la proposition est structurée comme suit :

- **corps du texte :** définition des règles de conception générale, règles de conception du schéma JSON et de la structure du schéma JSON, règles de conception des identificateurs du schéma JSON et de l'instance JSON;
- **annexe I :** règles de conversion des schémas XML de la norme ST.96 en schémas JSON, qui comprend l'appendice "Outil de conversion des définitions de schéma XML de la norme ST.96 en schémas JSON";
- **annexe II :** schémas JSON convertis à partir des schémas XML de la version 5.0 de la norme ST.96 de l'OMPI;
- **annexe III :** Exemples d'instances JSON correspondant aux exemples d'instances XML fournis dans l'annexe VII de la norme ST.96 de l'OMPI;
- **annexe IV :** acronymes et abréviations fondés sur la norme ST.96 de l'OMPI; et
- **annexe V :** termes de représentation fondés sur la norme ST.96 de l'OMPI.

12. Le Bureau international propose de donner le titre suivant à cette nouvelle norme de l'OMPI :

> "*Norme ST.97 de l'OMPI – Recommandation relative au traitement des données de propriété intellectuelle au format JSON (JavaScript Object Notation)*"

13. Le CWS est prié de noter que cette proposition concernant la norme JSON ne tient pas compte des questions relatives à l'architecture logicielle et à la langue d'application. La proposition de norme, ainsi que l'ensemble des annexes citées ci-dessus, sont reproduites dans l'annexe du présent document.

Travaux futurs

14. Dans le cadre de la préparation de la norme JSON, l'Équipe d'experts chargée de la norme XML4IP a relevé et pallié de nombreuses difficultés. Cependant, plusieurs problèmes encore en suspens devraient être résolus, tout comme il sera nécessaire d'apporter toute modification due à l'évolution de la spécification relative au schéma JSON, notamment :

- simplifier la structure des schémas JSON conformément à la pratique du secteur, par exemple, en supprimant les emboîtements inutiles dans les schémas JSON, qui ont découlé de la conversion à partir des schémas XML de la norme ST.96;

- améliorer la conception des données pour une validation plus précise des données, ce qui nécessitera l'analyse d'un spécialiste;
- actualiser l'ensemble des schémas JSON conformément aux modifications apportées à la version 6.0 de la norme ST.96, qui a ~~sera~~ été publiée en octobre 2022;
- ajouter des règles de conversion et des outils pertinents pour la conversion des instances XML en ~~format~~ JSON; et
- modifier la nouvelle norme, le cas échéant, pour tenir compte de toute modification apportée à la spécification relative au schéma JSON, que ce soit dans une nouvelle proposition ou une mise en service officielle.

MISE À JOUR DE LA TÂCHE N° 64

15. Lorsque le CWS aura adopté la proposition de nouvelle norme JSON, la tâche n° 64 sera considérée comme achevée et l'Équipe d'experts chargée de la norme XML4IP aura mené à bien ses travaux réalisés dans le cadre de cette tâche. Cependant, tel qu'indiqué précédemment, la norme devra faire l'objet de mises à jour à l'avenir. À cet égard, il est proposé d'actualiser la description de la tâche n° 64 comme suit :

"Procéder aux modifications et mises à jour nécessaires de la norme ST.97 de l'OMPI."

MISE À JOUR DE LA NOUVELLE NORME DE L'OMPI

16. L'Équipe d'experts chargée de la norme XML4IP ayant mené à bien ses travaux réalisés dans le cadre de la tâche n° 64 et élaboré la première version de la norme de l'OMPI relative au format JSON fondée sur les schémas XML de la norme ST.96 de l'OMPI, le Bureau international propose de confier à l'Équipe d'experts chargée des API la tâche n° 64 mise à jour, en parallèle de la gestion de la norme ST.90 de l'OMPI. Cela s'explique par le fait que le format JSON est très souvent utilisé en tant que charge utile pour les API RESTful.

17. Compte tenu des mises à jour dont la norme ST.96 de l'OMPI fait régulièrement l'objet, le CWS a mis en place une procédure accélérée pour l'examen et l'adoption des modifications apportées à la norme par l'Équipe d'experts chargée de la norme XML4IP. Alors que la nouvelle norme JSON devrait être mise à jour en permanence, parallèlement aux modifications de la norme ST.96 et à l'évolution de la spécification relative au schéma JSON, il est proposé de mettre en place une autre procédure accélérée pour l'examen et l'adoption des mises à jour apportées à la nouvelle norme JSON, comme suit :

a) toute proposition de mise à jour de la norme ST.97 de l'OMPI sera présentée directement à l'équipe d'experts ou transmise à l'équipe d'experts par l'intermédiaire du Secrétariat pour examen et approbation;

b) l'équipe d'experts désignée est provisoirement autorisée à adopter les mises à jour de la norme ST.97;

c) si une proposition de mise à jour de la norme ST.97 de l'OMPI pose problème, elle sera alors présentée au CWS pour examen, par exemple, lorsqu'il n'a pas été possible d'arriver à un consensus entre les membres de l'équipe d'experts désignée; et

d) le responsable de l'équipe d'experts désignée informera le CWS à sa prochaine session de toute mise à jour de la norme ST.97 de l'OMPI adoptée par l'équipe d'experts.

MODIFICATIONS D'ORDRE RÉDACTIONNEL APPORTÉES À LA NORME ST.90 DE L'OMPI

18. Aucun document n'étant présenté par l'Équipe d'experts chargée des API à la présente session, le Bureau international a pris note de plusieurs modifications d'ordre rédactionnel qu'il serait nécessaire d'apporter à la norme ST.90 de l'OMPI en cas d'adoption de la proposition

concernant la norme ST.97. Les modifications présentées au CWS pour examen sont les suivantes :

- ajout, dans la section Références, d'une référence à la nouvelle norme ST.97 de l'OMPI;
- modification du paragraphe 33 de la norme ST.90 de l'OMPI en vue d'introduire une référence à la norme ST.97 de l'OMPI par l'ajout d'une phrase soulignée dans le paragraphe, libellé comme suit :

  "Les API doivent prendre en charge les requêtes et les réponses aux formats XML et JSON. Pour le XML, les réponses doivent être conformes à une norme de l'OMPI utilisant le XML, comme la norme ST.96, et pour le JSON, les réponses doivent être conformes à la norme ST.97 de l'OMPI. Une correspondance systématique entre ces deux formats devrait être utilisée." ; et

- suppression de l'actuelle note de bas page n° 7 du paragraphe 33 de la norme ST.90, reproduite ci-après :

  "Une spécification JSON et un schéma JSON basés sur la norme ST.96 sont actuellement examinés par l'Équipe d'experts chargée de la norme XML4IP en vue de leur présentation pour examen à la huitième session du Comité des normes de l'OMPI qui se tiendra en novembre 2020, et de leur adoption en tant que nouvelle norme de l'OMPI. Entre-temps, la présente norme recommande la convention BadgerFish en raison de sa simplicité jusqu'à la mise à disposition du schéma JSON. Certains offices de propriété intellectuelle, comme l'OEB, la mentionnent également, www.epo.org/searching-for-patents/data/Webservices/ops.html."

19. Les propositions de modifications d'ordre rédactionnel citées au paragraphe 18 ci-dessus ne sont pas exhaustives. Il est donc suggéré que le Secrétariat introduise les modifications d'ordre rédactionnel nécessaires, le cas échéant, avant de publier à nouveau la nouvelle version de la norme ST.90 de l'OMPI.

20. Le CWS est invité

(a) à prendre note du contenu du présent document et de son annexe (proposition finale concernant la norme JSON),

(b) à examiner et à approuver le titre donné à la proposition de norme : "Norme ST.97 de l'OMPI – Recommandation relative au traitement des données de propriété intellectuelle au format JSON", tel qu'indiqué au paragraphe 12,

(c) à examiner et à adopter la nouvelle norme ST.97 de l'OMPI telle que reproduite dans l'annexe du présent document,

(d) à examiner et à approuver la révision de la description de la tâche n° 64 telle que figurant dans le paragraphe 15,

(e) à confier la tâche n° 64 à l'Équipe d'experts chargée des API, tel qu'indiqué au paragraphe 16,

(f) à examiner et à approuver la procédure accélérée aux fins de la mise à jour de la norme ST.97, tel qu'indiqué dans le paragraphe 17, et

(g) à approuver la modification de la norme ST.90 de l'OMPI en ce qui concerne la référence à la norme ST.97 adoptée et à prier le Secrétariat d'introduire toute autre modification d'ordre rédactionnel le cas échéant et de publier la norme ST.90 corrigée, tel qu'indiqué dans les paragraphes 18 et 19.

[L'annexe (proposition concernant la norme JSON) suit]

NORME ST.XX DE L'OMPI

RECOMMANDATION RELATIVE AU TRAITEMENT DES DONNÉES DE PROPRIÉTÉ INTELLECTUELLE AU FORMAT JSON

Proposition présentée par l'Équipe d'experts chargée de la norme XML4IP pour examen à la dixième session du Comité des normes de l'OMPI

Note de la rédaction : Pour faciliter la compréhension de certaines règles, des notes sont fournies dans l'ensemble de la proposition de projet de norme, mais elles seront retirées du document après l'adoption de la norme, à sa publication

TABLE DES MATIÈRES

TABLE DES MATIÈRES (TABLEAUX)

1. INTRODUCTION

La présente norme fournit des recommandations concernant la conception, la création et la mise à jour de ressources JavaScript Object Notation (JSON) qui seront utilisées pour le dépôt, le traitement, l'échange ou la publication de tout type de données de propriété intellectuelle. La présente norme examine également des règles relatives à la transformation des schémas XML (eXtensible Markup Language) (XSD) de la norme ST.96 de l'OMPI en schémas JSON qui respectent les recommandations susmentionnées.

La présente norme a pour but :

- de donner des orientations concernant la conception et l'élaboration de pratiques recommandées concernant les données de propriété intellectuelle au format JSON~~le balisage des données au format JSON~~;
- d'assurer la conformité au moyen de schémas et d'instances au format JSON fondés sur la norme ST.96 en ce qui concerne l'échange des données de propriété intellectuelle;
- de recommander des principes de conception relatifs à l'extension des schémas JSON élaborés ou à la création de nouveaux schémas JSON conformes; et
- de rationaliser l'échange de données en favorisant la réutilisation des ressources JSON entre les offices de propriété intellectuelle, ainsi que des données communiquées au public.

2. DÉFINITIONS ET TERMINOLOGIE

Aux fins de la présente norme, la terminologie suivante est utilisée :

- Le terme "ressources JSON" désigne toute composante utilisée la création ou la mise en œuvre d'une application JSON selon la présente norme;
- Les termes "`object`", "`object type`", "`property`", "`member`", "`property name`", "`property value`", "`property type`", "`keyword`" et "`definition`" utilisés dans la présente norme doivent être interprétés tels que définis dans la version du projet draft-2020-12[1] du JSON Schema Core;
- Le terme 'structure' utilisé dans la présente norme doit être interprété comme un 'socle' à partir duquel les schémas JSON sont construits;
- Le terme "définition générale" désigne une définition à laquelle d'autres définitions font référence dans le même schéma ou dans d'autres schémas; et
- Dans cette norme, les mots clés "DOIT", "DOIVENT", "NE DOIT PAS", "NE DOIVENT PAS", "DEVRAIENT", "DEVRAIENT", "NE DEVRAIT PAS", "NE DEVRAIENT PAS", "RECOMMANDÉ(S)", "PEUT", "PEUVENT", et "FACULTATIF(S)" doivent être interprétés comme indiqué dans l'appel à observation RFC 2119. Lorsque ces mots ne figurent pas en majuscules, ils doivent être pris dans leur sens courant en français.

3. NOTATIONS GÉNÉRALES

Les notations ci-après sont utilisées d'un bout à l'autre de la présente norme :

- <> : indique un terme utilisé pour décrire un espace qualifié qui, dans l'application, sera remplacé par une valeur d'instance spécifique;
- "" : indique que le texte entre guillemets doit être utilisé in extenso dans l'application;
- { } : indique que l'application est facultative; et
- Caractères `Courier New` : indique les mots clés JSON, les noms de propriété JSON et les éléments et attributs XSD.

3.1 Identificateurs de règle

Toutes les règles de conception sont normatives. Les règles de conception sont identifiées au moyen d'un préfixe de [JXX-nn].

- La valeur "JXX" est un préfixe qui sert à classer les types de règles comme suit :

  a) JGD pour les règles de conception générales;

---

[1] La version du schéma JSON est susceptible d'être modifiée, parce qu'elle n'a pas le statut RFC; elle n'a pas été adoptée par un groupe de travail IETF. Cette version de la norme se fonde sur la dernière version, c'est-à-dire la version 2020-12, disponible à l'adresse https://json-schema.org/draft/2020-12/json-schema-core.html.

b) JSD pour les règles de conception du schéma JSON;

c) JCD pour les règles de conception de la structure; et

d) JID pour les règles de conception des instances.

– La valeur “nn” définit le prochain numéro disponible dans l’ordre d’un type de règle donné. Il y a lieu de noter que le numéro ne désigne pas la position de la règle, en particulier pour une nouvelle règle. Une nouvelle règle sera placée dans le contexte pertinent. Par exemple, l’identificateur [JGD-10] identifie la dixième règle de conception générale. La règle [JGD-10] peut être placée entre les règles [JGD-05] et [JGD-06] au lieu de suivre [JGD-09] s’il s’agit de la position la plus appropriée pour cette règle.

– L’identificateur de la règle supprimée sera conservé tandis que la règle sera remplacée par le terme “Supprimé”.

3.2 Exemple de structure de données au format JSON

Des exemples de structure des données au format JSON sont présentés dans les encadrés dans une police à largeur fixe. Les exemples de syntaxe de la structure de données au format JSON sont surlignés pour en faciliter la lecture.

4. PORTÉE

La présente norme vise à apporter des ressources JSON à utiliser pour le dépôt, la publication, le traitement et le partage de données et d’informations en matière de propriété intellectuelle. La présente norme a pour objectif d’apporter des orientations aux offices de propriété intellectuelle et aux organisations concernées qui traitent des données et documents relatifs à des brevets, des marques, des dessins ou modèles industriels, des indications géographiques ou des œuvres orphelines protégées par le droit d’auteur.

La présente norme vise à apporter des orientations aux offices de propriété intellectuelle et aux organisations concernées qui créent ou modifient des données de propriété intellectuelle en tant que ressources JSON. Il est nécessaire de suivre cette norme pour assurer l’échange de données entre les offices de propriété intellectuelle qui utilisent des ~~messages~~ ressources JSON, telles que des schémas, des instances, des messages ou des charges utiles pour les interfaces de programmation d’application (API).

Les règles de conception et de conversion sont écrites en tenant compte des règles et conventions de conception de la norme ST.96 de l’OMPI. Les règles et conventions de conception de la norme ST.96 ne sont pas des mises en correspondance avec les règles et conventions de conception JSON et, par conséquent, les règles de conception ST.96 sont dupliquées et, dans certains cas, elles sont légèrement modifiées s’il y a lieu.

La présente norme comprend les annexes suivantes :

– Annexe I : Règles de conversion des schémas XML de la norme ST.96 en schémas JSON, qui comprend l’appendice : un outil de conversion des définitions de schéma XML de la norme ST.96 en schémas JSON;

– Annexe II : Schémas JSON convertis à partir des schémas XML de la version 5.0 de la norme ST.96 de l’OMPI[2];

– Annexe III : Exemples d’~~i~~instances JSON;

– Annexe IV : Liste des ~~A~~acronymes et abréviations; et

– Annexe V : Termes de représentation.

La présente norme ne tient pas compte :

a) des questions relatives à l’architecture logicielle; et

b) de la langue d’application.

5. RÈGLES DE CONCEPTION GÉNÉRALES AU FORMAT JSON

5.1 Aperçu

La présente section contient des règles et principes de conception JSON généraux et de haut niveau qui s’appliquent à tous les échanges de données JSON et à toutes les activités de développement JSON, plutôt qu’à un langage de programmation spécifique qui sérialise et désérialise les données vers ou à partir de JSON. Les règles et les principes

[2] Les schémas JSON convertis ont les mêmes noms de balise ainsi que la structure des données définie à l’annexe III de la norme ST.96, y compris les composantes XML à contenu mixte et les normes externes, c’est-à-dire MathML et Oasis Table, pour permettre l’interopérabilité avec les données en format ST.96.

généraux énumérés ci-dessous fournissent l'assise commune du schéma JSON, de l'instance JSON et de l'élaboration de la structure des données JSON pour toutes les données afin d'inclure les données liées ou non à la propriété intellectuelle, telles que les données des contenus mixtes. Les niveaux d'imbrication DEVRAIENT être limités au minimum lors de la création de nouveaux schémas JSON, de nouvelles instances JSON et de l'élaboration de structures de données JSON qui ne sont pas proposées dans la norme ST.96 de l'OMPI ou dont la compatibilité avec la norme ST.96 ne doit pas nécessairement être examinée.

5.2 Conventions de nommage JSON

Ces conventions sont nécessaires pour assurer la cohérence, l'uniformité et l'exhaustivité du nommage et de la définition de toutes les ressources JSON.

Ces conventions de nommage JSON reposent sur les lignes directrices et principes décrits dans le document ISO 11179, Partie 5 – Principes de dénomination et d'identification. Le nom des objets et les noms de propriété comprennent les termes suivants :

- La classe objet désigne une activité ou un objet dans un contexte d'entreprise et représente le regroupement ou l'agrégation logique de données (dans un modèle logique de données) auquel une propriété appartient. La classe objet est désignée par le terme de classe objet.
- Le terme de propriété détermine les caractéristiques de la classe objet.
- Le terme de qualification est constitué d'un ou plusieurs mots qui aident à définir un élément de donnée et à le différencier d'autres éléments de donnée connexes et qui peuvent être associés à un terme de classe objet ou à un terme de propriété si cela s'avère nécessaire pour rendre un nom unique.
- Le terme de représentation classe le format de l'élément de donnée selon de grandes catégories. Les termes de représentation énumérés à l'annexe V devraient être utilisés.

[JGD-01] Le type d'objet et les noms de propriété DOIVENT être composés de mots anglais, écrits selon l'orthographe anglaise figurant dans le dictionnaire Oxford English Dictionary. Les seules exceptions acceptées sont les sigles, les abréviations ou d'autres mots tronqués qui sont énumérés à l'annexe IV.

[JGD-02] Le type d'objet et les noms de propriété DEVRAIENT comprendre uniquement des noms, des adjectifs et des verbes au présent, à l'exception des sigles, des abréviations et d'autres mots tronqués qui sont énumérés à l'annexe IV.

[JGD-03] Les caractères utilisés dans les noms de propriété DOIVENT être limités à la série suivante : 'a-z, A-Z et 0-9'.

[JGD-04] La longueur maximale d'un type d'objet et des noms de propriété NE DEVRAIT PAS être supérieure à 35 caractères.

[JGD-05] Le type d'objet et les noms de propriété DEVRAIENT être concis et s'expliquer d'eux-mêmes.

[JGD-06] Le type d'objet et les noms de propriété DOIVENT figurer en caractères minuscules de type "camel" (LCC). Par exemple, "`currencyCode`":"EUR".

[JGD-07] Les noms de type d'objet DOIVENT figurer en caractères minuscules de type "camel" (LCC) et avoir un type suffixe. Par exemple, `applicantType`.

[JGD-08] Les sigles et les abréviations énumérés à l'annexe IV DOIVENT toujours être utilisés en lieu et place du nom développé complet.

[JGD-09] Les sigles et abréviations DOIVENT figurer comme indiqué à l'annexe IV pour les noms de propriété et de types d'objet.

[JGD-10] Un terme de classe objet DOIT toujours avoir la même signification sémantique dans un domaine spécifique de la propriété intellectuelle, tel que les brevets, les marques, les dessins et modèles industriels, les indications géographiques ou le droit d'auteur, et PEUT comprendre plus d'un mot. Par exemple, `contactInformation`.

[Note : l'expression "espace de nommage" utilisée dans la norme ST.96 a été remplacée par l'expression "domaine de la propriété intellectuelle" dans JSON]

[JGD-11] Le terme de propriété dans un nom DOIT être unique dans le contexte d'une classe objet mais PEUT être réutilisé dans différentes classes objets.

[JGD-12] Un terme de qualification PEUT être attaché à un terme de classe objet ou à un terme de propriété si cela s'avère nécessaire pour rendre un nom unique.

[JGD-13] Lorsqu'un nom comprend un terme de classe objet, un terme de propriété et un terme de représentation, le terme de classe objet DOIT précéder le terme de propriété et le terme de propriété DOIT précéder le terme de représentation. Un terme de qualification DEVRAIT précéder le terme associé de classe objet ou le terme de propriété. Par exemple, `claimTotalQuantity`.

[JGD-14] Si le terme de propriété finit avec le même mot que le terme de représentation (ou un mot équivalent), le terme de représentation DOIT être supprimé.

[JGD-15] Lorsqu'un terme de représentation est requis, les termes de représentation de l'annexe V DOIVENT être utilisés pour les termes de représentation dans les noms de composantes basiques.

[JGD-16] Dans un domaine de la propriété intellectuelle, tous les types d'objet et les noms de propriété DOIVENT être uniques.

[JGD-17] Le ou les mots dans un nom DEVRAIENT être utilisés au singulier sauf si le concept est lui-même au pluriel. Par exemple, `totalMarkSeries`.

[JGD-18] Le nom d'une propriété ou d'un type d'objet qui comprend une collection de composantes contextuellement reliées DEVRAIT avoir le suffixe "Bag". Par exemple, `emailAddressBag` représente une collection d'éléments `emailAddress`.

[JGD-19] Des mots qui relient comme "and", "of" et "the" NE DEVRAIENT PAS être utilisés dans les types d'objets et les noms de propriété à moins qu'ils ne fassent partie de la terminologie commerciale.

[JGD-20] Le type d'objet et les noms de propriété NE DOIVENT PAS être traduits, modifiés ou remplacés dans un but, quel qu'il soit.

[JGD-21] Le type d'objet et les noms de propriété NE DOIVENT PAS faire référence à des numéros d'article et de règle. Par exemple, `PCTRule702C` pour le PCT.

[JGD-22] Les niveaux d'imbrication DEVRAIENT être limités au minimum lors de la création de nouveaux schémas JSON, d'instances JSON et de structures/fragments de données JSON qui ne figurent pas dans la norme ST.96 de l'OMPI.

[JGD-23] Pour de nouveaux types d'objet et noms de propriété qui ne sont pas définis dans la norme ST.96 de l'OMPI ou ne sont probablement pas compatibles avec un nom de composante qui sera défini dans la norme ST.96, des termes ou des noms descriptifs inclus (*inline*) DEVRAIENT être utilisés au lieu de types d'objet ou de noms de propriété courts ou génériques ou qui sont des objets ayant une seule propriété. Par exemple, on préférera "`inventorFullName`": "`Thomas Edison`" à "`inventor`": { "`fullName`": "`Thomas Edison`" }.

[Notes : Cette règle recommande de suivre les pratiques en vigueur dans le secteur pour le format JSON, dans le cas où les noms ne sont pas liés aux données XML reposant sur la norme ST.96 actuelle ou à venir.]

6. RÈGLES DE CONCEPTION DU SCHÉMA JSON

6.1 Aperçu

Le schéma JSON décrit la structure d'une instance JSON, qui exprime les contraintes sur la structure et le contenu du document. La présente norme devrait être conforme à la spécification du schéma JSON du secteur. La dernière version disponible au moment de la publication de la présente norme est le projet 2020-12 et la présente version de la norme se réfère à ce projet de spécification.

[JSD-01] Les schémas JSON DOIVENT se conformer aux spécifications du schéma JSON : la version 2020-12 de JSON Schema Core, disponible à l'adresse suivante : https://json-schema.org/latest/json-schema-core.html, et la version 2020-12 de JSON Schema Validation, disponible à l'adresse : https://json-schema.org/latest/json-schema-validation.html.

[JSD-02] Les schémas JSON DOIVENT indiquer qu'ils se conforment à la version 2020-12 du schéma JSON à l'aide du mot clé `$schema` et de la valeur "https://json-schema.org/draft/2020-12/schema".

Exemple : Indiquer la version du schéma JSON
`"$schema": "https://json-schema.org/draft/2020-12/schema"`

Le schéma devrait être codé en UTF-8 pour garantir une interopérabilité maximale.

[JSD-03] Les schémas JSON DOIVENT utiliser la norme ISO/IEC 10646 – UCS – ensemble de caractères Unicode. Le format UTF-8 DOIT être utilisé pour coder les caractères Unicode.

6.2 Modularité

La modularité permet la création de composantes de schéma pour favoriser la flexibilité dans la conception et la réutilisabilité. Dans la conception, il est recommandé d'éviter la définition de toutes les propriétés et composantes logiques sous la forme d'un seul schéma JSON monolithique, qui empêche de partager et de réutiliser des propriétés individuelles ou des composantes logiques définies comme étant un groupe dans un schéma.

Le schéma présenté ci-après ne respecte **pas** le principe de modularité. Il N'EST PAS recommandé par la présente norme.

applicationNumber.json (Exemple de document de schéma composé incorrect)

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

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

```
        } ]
      }
    }
}
```

L'approche de conception préférée consiste à séparer les données en groupe de petites composantes représentées par les modules de schéma, présentés dans le nouveau schéma de numéro de demande ci-après. Ce schéma JSON repose sur de plus petits modules de schéma JSON définis individuellement dans leurs propres schémas.

applicationNumber.json (Exemple de schéma modulaire)

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

applicationNumberType.json (Exemple de schéma modulaire – suite)

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

ipOfficeCode.json (Exemple de schéma modulaire – suite)

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

extendedWIPOST3CodeType.json (Exemple de schéma modulaire – suite)

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

wipoST3CodeType.json (Exemple de schéma modulaire – suite)

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

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

wipoFormerST3CodeType.json (Exemple de schéma modulaire – suite)

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

st13ApplicationNumber.json (Exemple de schéma modulaire – suite)

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

st13ApplicationNumberType.json (Exemple de schéma modulaire – suite)

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

applicationNumberText.json (Exemple de schéma modulaire – suite)

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

Les schémas JSON devraient utiliser le mot clé "`$defs`" pour créer des définitions générales pour les propriétés et leurs contenus qui peuvent être réutilisés, comme l'illustre l'exemple ci-dessus. Cela correspond à peu près à la création de déclarations générales d'éléments et de types nommés dans un schéma XML.

[JSD-04] Les schémas JSON DEVRAIENT utiliser le mot clé "`$defs`" qui inclut une définition réutilisable pour chaque propriété ou type de propriété.

[JSD-05] Les développeurs DOIVENT utiliser les schémas JSON existants qui sont définis à l'annexe II de la présente proposition de norme, le cas échéant, avant d'en créer de nouveaux.

[JSD-06] Les développeurs DEVRAIENT créer de nouveaux schémas JSON seulement après avoir déterminé qu'aucun schéma JSON existant ne décrivait de manière adéquate la structure en question.

6.3 Documentation

Les schémas JSON devraient être autodescriptifs. Les développeurs devraient viser à concevoir des noms de structure JSON qui ont un sens. En outre, le schéma JSON devrait avoir une documentation décrivant le schéma et les structures JSON.

Pour favoriser la réutilisabilité en restant général, le schéma JSON ne devrait pas fournir de documentation sur des détails de mise en œuvre relatifs à un système spécifique.

[JSD-07] La documentation NE DEVRAIT PAS décrire des détails ou d'autres informations qui ne sont pas directement liés à la signification de la structure.

Un en-tête de schéma JSON permet à un développeur de schémas de discerner facilement le but, l'utilisation et le contenu d'un schéma. Cette information est très utile lorsqu'un développeur doit choisir un schéma à utiliser comme canevas pour la création d'un autre schéma.

[JSD-08] Les schémas JSON DEVRAIENT inclure la documentation relative à l'en-tête du schéma JSON en utilisant le mot clé "`description`".

[JSD-09] Les éléments énumérés dans le tableau 1 ci-après DEVRAIENT être inclus dans l'en-tête de tous les schémas JSON.

Tableau 1 : Éléments de la documentation relative à l'en-tête du schéma JSON

Nom de l'en-tête	Description	Requis/Facultatif
`Description`	Description textuelle claire de l'information décrite par le schéma.	Requis, sauf pour les schémas JSON provenant de types simples de définitions de schémas XML de la norme ST.96 pour lesquels une description n'est pas fournie, tels que `DateType`.
`Version`	Nombre de versions majeures et mineurs du schéma	Requis
`SchemaCreatedDate`	Date de création du schéma	Facultatif
`SchemaLastModifiedDate`	Date de la dernière modification du schéma	Facultatif
`SchemaContactPoint`	Nom de l'organisation à contacter pour toute question relative au schéma	Facultatif
`SchemaReleaseNoteURL`	Emplacement où les notes de diffusion du schéma sont publiées	Facultatif

[JSD-10] Les éléments sur la documentation relative à l'en-tête, tels que Publié le et le nombre de versions ci-dessus DEVRAIENT être séparés par des points-virgules, avec des espaces possibles après le point-virgule, correspondre à la valeur associée au mot clé "`description`". Si une valeur n'est pas disponible pour l'élément de l'en-tête, alors seule l'étiquette sera incluse, comme l'illustre l'exemple ci-après :

Exemple de documentation relative à l'en-tête pour applicationBody (Schéma au niveau du document)
"description" : "Description: Body of a patent application; Version: V5_0; SchemaCreatedDate: 2012-07-13; SchemaLastModifiedDate: 2021-10-01; SchemaContactPoint: xml.standards@wipo.int; SchemaReleaseNoteURL: http://www.wipo.int/standards/XMLSchema/ST96/V5_0/ReleaseNotes.pdf"

Exemple de documentation relative à l'en-tête pour ipOfficeCode (Schéma pas au niveau du document)
"description" : " Description: Two-letter alphabetic codes which represent the names of states, other entities and intergovernmental organizations the legislation of which provides for the protection of IP rights or which organizations are acting in the framework of a treaty in the field of IP; Version: V5_0"

Exemple de documentation relative à l'en-tête pour appellateBodyCategoryType.json (Schéma de définitions de type énumération)
"description" : "Version: V5_0; Office appeal board: Appeal board within the IP office; Court: Court; Appeal Court: Second instance court; Supreme Court: Highest appellate court"

Exemple de documentation relative à l'en-tête pour un schéma de définition de type
"description" : "Version: V5_0"

6.4 Nom du fichier

Le nom de fichier d'un schéma JSON suit les règles de la norme ST.96, si ce n'est qu'il doit figurer en LCC.

Les noms de fichier d'un schéma et les noms d'un schéma sont souvent associés. Les noms de fichier d'un schéma reposent sur les noms de schéma correspondants. C'est ainsi, par exemple, que le nom de fichier `postalAddressType.json` vient du nom de schéma `postalAddressType`. Par conséquent, les conventions de nommage des fichiers de schéma sont liées aux règles des conventions de nommage JSON dans la présente norme.

Un fichier de schéma PEUT contenir les informations de la version. Un schéma qui en est au stade de projet peut être révisé. Les projets de schéma doivent être indiqués comme tels dans le nom de fichier du schéma, au moyen de la lettre "D" et du numéro de la révision.

[JSD-11] Les caractères utilisés dans les noms de fichier de schéma DOIVENT appartenir à la série suivante : 'a-z, A-Z, 0-9, caractère de soulignement "_", et point "."'.

[JSD-12] Un nom de fichier de schéma DOIT comporter deux parties obligatoires avec un délimiteur et les informations facultatives de la version avec deux délimiteurs, c'est-à-dire : `<component name>{"_"V"<major version number>"_"<minor version number>"}."<file extension>;`. Par exemple, `emailAddressType.json`, `languageCode.json`, `applicationBody_V1_0.json`.

[JSD-13] Un nom de fichier du projet de schéma DOIT comporter quatre parties obligatoires avec deux délimiteurs et les informations facultatives de la version avec deux délimiteurs supplémentaires, c'est-à-dire : `<component name>{"_"V"<major version number>"_"<minor version number>"}_"D"<revision number>"."<file extension>`, par exemple, `trademarkApplication_V1_1_D1.json`. Si un projet de schéma est fondé sur un schéma existant et que son nom de fichier inclut les informations de la version, les numéros des versions majeure et mineure dans le nom de fichier du projet de schéma DEVRAIENT être ceux qui sont indiqués dans le fichier de schéma sur lequel est fondé le projet de schéma. Si le projet de schéma est nouveau, le numéro de la version majeure dans le nom de fichier du projet de schéma DEVRAIT être celui qui est indiqué dans le domaine de la propriété intellectuelle correspondant et un numéro de version mineure dans le fichier du projet de schéma DEVRAIT être zéro "0".

6.5 Structure des propriétés du schéma JSON

Les schémas JSON devraient avoir la propriété `"type": "object"` afin de garantir que le format JSON soit utilisé uniquement pour les structures imbriquées et non pour les valeurs individuelles. À partir de l'exemple `applicationNumber.json` ci-après :

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

[JSD-14] L'objet du schéma le plus extérieur DOIT comporter un mot clé "`type`" dont la valeur est "`object`".

[JSD-15] L'objet du schéma le plus extérieur DOIT comporter un mot clé "`$defs`" dont la valeur est la propriété de l'objet du schéma le plus extérieur.

[JSD-16] L'objet du schéma le plus extérieur DOIT comporter un mot clé "`required`" dont la valeur est un tableau qui contient un seul élément, par exemple, la propriété de l'objet du schéma le plus extérieur.

Les extensions de schéma JSON (personnalisations) peuvent être utilisées. Si le type étendu est un type objet, il convient de le référencer au lieu de dupliquer ses propriétés pour favoriser la réutilisabilité.

[JSD-17] Les extensions de schéma JSON (personnalisations) pour les types d'objet DOIVENT être appliquées en renvoyant au schéma JSON du type étendu.

7. RÈGLES DE CONCEPTION DES STRUCTURES DU SCHÉMA JSON

7.1 Aperçu

La présente section établit les règles relatives aux structures du schéma JSON, en particulier les tableaux, les objets et les valeurs primitives. La normalisation des noms relatifs aux structures du schéma est essentielle à l'élaboration d'une architecture de données solide.

7.2 Propriétés

Les propriétés, aussi appelées membres, sont le socle d'une structure JSON.

[JSC-01] Les définitions DEVRAIENT utiliser dans toute la mesure du possible des schémas existants.

[JSC-02] Les propriétés multiples qui peuvent logiquement être groupées PEUVENT être déclarées dans un seul fichier de schéma dans la définition générale.

7.3 Définitions

Chaque propriété devrait avoir une définition générale qui est définie dans son schéma JSON. Cela permettra de réutiliser le nom de propriété dans de nombreux parents et d'avoir une définition cohérente pour l'ensemble d'entre eux. Veuillez consulter l'exemple de la propriété "`applicationNumber`" ci-après.

[JSC-03] Chaque propriété énumérée dans un mot clé de propriété DEVRAIT renvoyer à une définition générale qui est définie dans le mot clé "`$defs`". La définition générale DEVRAIT avoir le même nom que la propriété.

Exemple de propriété renvoyant à une définition générale

```
{
  "$id" : "applicationNumber.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "applicationNumber" : {
      "$ref" : "#/$defs/applicationNumber"
    }
  },
```

```
  "required" : [ "applicationNumber" ],
  "$defs" : {
    "applicationNumber" : {
      "$ref" : "applicationNumberType.json#/$defs/applicationNumberType",
      "description" : "Description: Numbers used by IPOs in order to identify each
application received; Version: V5_0"
    }
  }
}
```

La définition générale d'une propriété devrait comprendre le nom de fichier et la description de la propriété.

[JSC-04] La définition générale d'une propriété DEVRAIT comprendre le nom de fichier et la "description" de la propriété.

Les propriétés doivent avoir des types. Ils peuvent être définis directement s'il s'agit de types primitifs (à l'exception des objets) ou renvoyer à une définition générale de propriété dans un autre schéma JSON.

[JSC-05] Une propriété DOIT avoir un type qui est spécifié en utilisant le mot clé "type", soit comme propriété directe soit par renvoi à une définition générale.

7.4 Définitions de types

Les schémas JSON peuvent définir des définitions de type réutilisables auxquelles renvoient les définitions générales de propriété. Ces définitions générales de types devraient comprendre un mot clé "type", un mot clé "properties" (si le type est "object") et toute autre contrainte en matière de valeur.

[JSC-06] Un schéma PEUT définir des définitions générales de type afin de réutiliser les modèles de contenu pour de nombreuses propriétés.

Définition de type réutilisable

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

[JSC-07] Les définitions qui représentent des types DOIVENT avoir des noms qui figurent sous la forme LCC + suffixe "type".

7.5 Type primitif JSON

[JSC-08] Le type primitif JSON pertinent le plus précis DEVRAIT être utilisé pour une propriété. Les types primitifs comprennent : "string", "number", "integer", "object", "array", "boolean", et "null".

Par exemple, si une valeur de propriété est un nombre entier, le type "integer" devrait être préféré au type "number", qui est plus générique, ou au type "string", qui est plus permissif. Pour le type "string", les formats incorporés devraient être utilisés s'il y a lieu, par exemple "date-time" ou "duration".

7.6 Listes de codes

Dans certains cas, il est intéressant de limiter une valeur à une liste de codes normalisés et acceptables à des fins d'échange de données. Les listes de codes sont un moyen de créer un vocabulaire contrôlé de valeurs autorisées pour un élément de donnée (par exemple, une liste de codes normalisés pour les codes de noms de pays, les codes de

langue, les codes d'office de propriété intellectuelle, etc.). Les listes de codes qui existent déjà dans le domaine public et dont la gestion est assurée par des comités des normes compétents tels que l'ISO devraient être utilisées.

[JSC-09] La norme ST.3 de l'OMPI DOIT être utilisée pour la représentation des offices de propriété intellectuelle, des États, d'autres entités, d'organisations, ainsi que pour les pays/organisations désignés et prioritaires.

[JSC-10] La norme ISO 3166-1-Alpha-2 Éléments de code (codes de pays à deux lettres) DOIT être utilisée pour la représentation des noms de pays dans les informations relatives à l'adresse ou à la nationalité.

[JSC-11] La norme ISO 639-1 (codes de langue à deux lettres) DOIT être utilisée pour les codes de langue.

[JSC-12] La norme ISO 4217-Alpha (codes pour la représentation des monnaies à trois lettres) DOIT être utilisée pour les codes des monnaies.

[JSC-13] Le mot clé JSON `enum` DEVRAIT être utilisé pour définir les listes de codes.

[JSC-14] Les caractères utilisés dans les valeurs d'énumération DOIVENT se limiter aux caractères suivants : `{a-z, A-Z, 0-9`, point (.), virgule (,), espaces, tiret (-) and caractère de soulignement (_)}

7.7 Tableaux

Le terme cardinalité est défini comme étant le nombre d'éléments dans un tableau. La cardinalité est indiquée dans un schéma à l'aide des mots clés `minItems` et `maxItems`. Il est recommandé que les développeurs de schémas ne précisent pas de valeurs implicites pour les indicateurs d'occurrence (c'est-à-dire "`minItems`": 0) parce que cela pourrait encombrer inutilement un schéma.

[JSC-15] Les schémas JSON DEVRAIENT utiliser les mots clés `minItems` et `maxItems` pour les tableaux, sauf pour la valeur implicite de `minItems` (0).

Le type d'élément dans un tableau doit être défini en utilisant le mot clé "`items`". Par souci de simplicité, tous les éléments d'un tableau doivent avoir le même type. Si on souhaite avoir une séquence d'objets de différents types, ces types doivent être définis en tant que propriétés distinctes d'un objet.

[JSC-16] Pour chaque objet de type tableau, il DOIT y avoir un mot clé "`items`" et sa valeur DOIT être un seul objet de schéma et non un tableau. Tous les éléments d'un tableau DOIVENT avoir le même type.

Le mot clé "`additionalItems`" ne doit pas être utilisé pour des tableaux, car il n'est pas pertinent lorsque la valeur des "`items`" est un seul objet de schéma.

[JSC-17] Le mot clé "`additionalItems`" NE DEVRAIT PAS être utilisé quand "`items`" est un seul objet de schéma.

7.8 Objets

7.8.1 "Caractères génériques" en matière de propriété

Les schémas JSON ne devraient pas permettre à des propriétés arbitraires d'être incluses dans l'instance JSON tout en restant valables, car cela pourrait nuire à l'intégrité de l'échange de données.

Le mot clé "`additionalProperties`" doit être utilisé avec la valeur "`false`". À défaut, des propriétés non définies seront autorisées dans les instances.

[JSC-18] Un schéma JSON DOIT utiliser "`additionalProperties`" avec la valeur "`false`" pour chaque objet.

L'utilisation du mot clé "`patternProperties`" n'est pas autorisée. Ce mot clé permet de mettre en correspondance des expressions régulières et des schémas. Par exemple, elle autorise des définitions de schéma implicites reposant sur un nom de propriété. À partir d'un type de nom de propriété particulier, un schéma particulier est appliqué.

[JSC-19] Un schéma NE DOIT PAS utiliser le mot clé "`patternProperties`".

7.8.2 Ordre des propriétés

Le schéma JSON n'applique pas d'ordre particulier pour les propriétés d'un objet. Cependant, si le schéma JSON a un schéma XML correspondant, il est recommandé d'énumérer les propriétés dans le même ordre que dans le schéma XML, aussi bien dans le schéma que dans l'instance JSON.

[JSC-20] Un schéma DEVRAIT utiliser le même ordre de propriété que dans le schéma XML correspondant, s'il en existe un.

8 IDENTIFICATEURS DE SCHÉMA JSON

8.1 Aperçu

Un ID dans un schéma JSON fournit un URI qui identifie une catégorie d'information reposant sur un domaine commercial (par exemple, entreprise, brevets, et marques). La présente proposition de norme a choisi d'utiliser des relations plusieurs-à-un entre les quelques ID et les centaines de structures JSON. Un groupe de structures JSON connexes avec des noms uniques sera associé à un ID particulier. Un uniform resource identifier (URI) devrait être utilisé pour l'identification.

[JID-01] Les ID DOIVENT être utilisés dans les schémas à l'aide du mot clé "`$id`".

9. RÈGLES DE CONCEPTION DES INSTANCES JSON

Le schéma JSON définit la structure et les contraintes pour l'instance JSON. Pour accroître les échanges de données (intra et inter offices) et en assurer la qualité, les instances JSON devraient être associées au schéma JSON pour en garantir la validité et la conformité des instances.

9.1 Ordre des propriétés

Le schéma JSON n'applique pas d'ordre particulier pour les propriétés d'un objet. Cependant, il est recommandé que les propriétés figurent dans le même ordre dans l'instance et dans le schéma JSON.

[JIN-01] Un document d'instance JSON DEVRAIT utiliser le même ordre de propriétés que dans le schéma JSON correspondant, s'il en existe un.

9.2 Validation de l'instance JSON

La validation d'instances JSON garantit que leur contenu remplisse toutes les exigences définies dans les schémas correspondants.

[JIN-02] Les documents d'instance JSON PEUVENT être validés au regard d'un schéma correspondant durant le traitement.

[JIN-03] Un schéma d'exécution PEUT être créé pour remplir les exigences en matière de performance de l'application dans l'environnement d'exécution. Par exemple, le schéma composé pour le numéro de demande peut être utilisé comme schéma d'exécution.

[Note : Le schéma d'exécution JSON est semblable au schéma mis à plat de la norme ST.96 utilisé dans l'environnement de production notamment pour des questions d'efficacité. Nous pensons donc que le schéma d'exécution JSON remplirait le même objectif. Cependant, au moment de l'élaboration de la présente norme, aucune pratique ou assistance en ce sens n'est proposée par les outils/éditeurs dans le secteur JSON.]

[JIN-04] Toute modification, mise à jour, révision et nouvelle diffusion DOIT d'abord être soumise à l'Équipe d'experts chargée de la norme XML4IP pour approbation avant que les changements puissent être incorporés au schéma d'exécution.

Il est souhaitable qu'une instance JSON soit conforme à un schéma.

[JIN-05] Une instance JSON DEVRAIT être conforme à un schéma JSON particulier, lui-même conforme aux règles décrites dans la présente norme.

10. RÉFÉRENCES

Normes de l'OMPI

- Norme ST.96 de l'OMPI : Utilisation du XML dans le traitement de l'information en matière de propriété intellectuelle
- Norme ST.90 de l'OMPI: Recommandations relatives au traitement et à la communication des données de propriété intellectuelle aux API Web

Normes et spécifications industrielles

- Spécification JSON : http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
- JSON Schema Core, projet 2020-12 : http://json-schema.org/latest/json-schema-core.html
- JSON Schema Validation, projet 2020-12 : http://json-schema.org/latest/json-schema-validation.html
- Spécification Open API v3.1.0 : https://spec.openapis.org/oas/v3.1.0#schema
- Norme ISO 21778 ECMA-404 –

	Syntaxe d’échange de données JSON:	https://www.ecma-international.org/publications/standards/Ecma-404.htm
–	UBL-2.1 Langage d’affaires universel:	http://docs.oasis-open.org/ubl/UBL-2.1.html
–	Norme ISO 3166 Codes des noms de pays :	https://www.iso.org/fr/iso-3166-country-codes.html
–	Norme ISO 639 Codes langues :	https://www.iso.org/fr/iso-639-language-codes.html
–	Norme ISO 4217 Codes des monnaies :	https://www.iso.org/fr/iso-4217-currency-codes.html
–	Norme ISO 11179 :	https://www.iso.org/fr/standard/60341.html
–	Appel à observations 2119 :	https://www.ietf.org/rfc/rfc2119.txt

[L’annexe I suit]

ST.XX – ANNEXE I

RÈGLES DE CONVERSION DES SCHÉMAS XML DE LA NORME ST.96 EN SCHÉMAS JSON ET PRINCIPES D'UTILISATION

INTRODUCTION

Le format JSON est progressivement adopté et utilisé par les offices de propriété intellectuelle et le secteur de la propriété intellectuelle tandis que le format XML (eXtensible Markup Language) reposant sur les normes XML de l'OMPI est encore largement utilisé. La norme ST.96 de l'OMPI recommande d'utiliser les ressources XML pour le dépôt, la publication, le traitement et l'échange de divers ~~tout~~ types de données sur la propriété intellectuelle, c'est-à-dire les brevets, les marques, les dessins et modèles industriels, les indications géographiques et le droit d'auteur. La norme ST.96 est appliquée par les offices de propriété intellectuelle telle que publiée ou elle est personnalisée s'il y a lieu.

Pour faciliter l'échange de données entre les offices de propriété intellectuelle et la diffusion de données par les offices de propriété intellectuelle dans deux formats, à savoir XML et JSON, la cohérence et la compatibilité des structures des données et des noms de balise entre les deux formats sont importantes. Cette cohérence et cette compatibilité des données peuvent être assurées à l'aide des schémas XML et des schémas JSON compatibles qui seront utilisés pour valider les instances XML et les instances JSON, respectivement.

L'annexe I a pour but de fournir des règles et des principes de conversion des définitions de schéma XML de la norme ST.96 en schémas JSON correspondants. Les schémas JSON qui figurent à l'annexe II de la norme ST.XX sont le résultat de l'application de ces règles de conversion et il est recommandé aux offices de propriété intellectuelle de suivre ces règles lors de l'élaboration de schémas JSON, qui seront cohérents et compatibles avec leurs définitions de schéma XML de la norme ST.96 personnalisées.

L'annexe I comprend également un outil de conversion, qui figure en appendice, qui applique ces règles pour utilisation par les offices de propriété intellectuelle souhaitant élaborer leurs propres schémas JSON en se fondant sur leurs définitions de schéma XML de la norme ST.96 personnalisées.

Au moment de l'élaboration du présent document, le projet 2020-12 est la dernière version de la proposition de schéma JSON et la version 5.0 est la dernière version de la norme ST.96. C'est pourquoi les règles et principes de conversion s'appuient sur le projet JSON 2020-12 et sur la version 5.0 de la norme ST.96, qui utilise la version 1.1 des définitions de schéma XML. Les règles fournies devraient être conformes aux règles définies dans le corps de la norme ST.XX. Les règles de conversion sont identifiées dans le présent annexe au moyen du préfixe 'TR'.

ALGORITHME ET PRINCIPES DE CONVERSION

Pour convertir les définitions de schéma XML de la norme ST.96 en schéma JSON, il convient d'appliquer l'algorithme suivant :

1. Convertir le nom du fichier XSD en nom de fichier de schéma JSON; voir les règles relatives au nom de fichier à la section **Error! Reference source not found.**

2. Convertir un fragment XSD en fragment de schéma JSON compatible ~~à l'aide :~~

   ~~a) d'expressions XPaths absolues; ou~~

   ~~b) d'expressions Xpaths relatives.~~

Il convient de noter que la conversion des définitions de schéma XML en schémas JSON n'exige aucun ordre strict. Cela étant, la validation des schémas JSON obtenus devrait être générée une fois que l'ensemble des schémas ont été créés, parce qu'un schéma JSON obtenu peut avoir des dépendances "`$ref`" qui n'ont pas encore été créées.

PORTÉE

Comme les schémas JSON publiés dans la présente norme résultent de la conversion des définitions de schéma XML de la norme ST.96, la présente annexe I couvre uniquement les constructeurs, les mots clés et autres éléments relatifs aux définitions de schéma XML utilisés dans la norme ST.96 de l'OMPI.

CONVERSION DE NOM DE FICHIER

[TR-01] Les noms de fichier de schéma JSON et les noms d'objet ou de propriété de schéma JSON DOIVENT être les mêmes que les noms de fichier ou les noms composantes des définitions de schéma XML de la norme ST.96 correspondants, mais ils doivent respecter les règles définies à la section **Error! Reference source not found.** "Conventions de nommage JSON de la présente norme" et utiliser le suffixe ".json" au lieu du suffixe ".xsd" pour les noms de fichier.

CONVERSION DES ESPACES DE NOMMAGE

Les espaces de nommage XSD ne devraient pas être conservés dans les schémas JSON. Il convient de noter que la caractéristique espace de nommage natif n'est pas prise en charge dans le schéma JSON 2022-12. Si, à l'avenir, cette caractéristique est prise en charge par une norme relative au schéma JSON, la conversion de l'espace de nommage XSD sera réexaminée. Cependant, l'imitation de l'espace de nommage ne peut être réalisée que si les schémas JSON sont stockés dans des sous-dossiers par domaine de propriété intellectuelle. La conception de la norme ST.96 prévoit des sous-dossiers séparés pour chaque domaine de propriété intellectuelle.

[TR-02] Pour l'émulation des espaces de nommage des définitions de schéma XML, les schémas JSON DOIVENT être stockés dans des sous-dossiers par domaine de propriété intellectuelle et dans le domaine commun, c'est-à-dire le domaine commun ou les domaines de droit d'auteur, de dessins et modèles, d'indications géographiques, de brevet, de marques et de normes externes, de la même manière que les XSD correspondants sont stockés dans la norme ST.96.

CONVERSION DES TYPES DE DONNÉES INCORPORÉES

Selon la règle JSC-09, le type primitif JSON pertinent le plus spécifique DEVRAIT être utilisé pour une propriété. Par exemple, si une valeur de propriété est un nombre entier, le type "`integer`" doit être préféré au type "`number`", qui est plus générique, ou au type "`string`", qui est plus permissif.

Le tableau 2 fournit une mise en correspondance d'un type de données XSD avec un type de données JSON (incorporée ou personnalisée/incluse ou dans un fichier séparé) utilisés pour la conversion de définitions de schéma XML de la norme ST.96 en schéma JSON.

[TR-03] Les types de données XSD énumérés dans le tableau suivant DEVRAIENT être convertis en types de données du schéma JSON correspondant, comme défini dans le tableau 2.

Tableau 2 : Conversion des types de données XSD simples

Type de données XSD	**Type de données du schéma JSON ou définition du type de schéma JSON**
`xsd:string`	`"type": "string"`
`Token`	`"type": "string"`
`xsd:integer`	`"type": "integer"`

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

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

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

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

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

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

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

`gYearMonth`	Il n'existe aucun type incorporé. Un fichier spécifique "gYearMonth.json" est donc défini.
	`"gYearMonth": {`
	`"anyOf": [`
	`{`
	`"type": "object",`
	`"properties": {`
	`"year": {`
	`"type": "integer"`
	`},`
	`"month": {`
	`"type": "integer",`
	`"minimum": 1,`
	`"maximum": 12`
	`},`
	`"timezone": {`

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

gYear	Il n'existe aucun type incorporé. Un fichier spécifique "gYear.json" est donc défini.
	"gYear": {
	"anyOf": [
	{
	"type": "object",
	"properties": {
	"year": {
	"type": "integer"
	},
	"timezone": {
	"type": "integer",
	"minimum": -1440,
	"maximum": 1439
	}
	}
	}
	]
	}

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

DÉFINITION XSD

Pour convertir une définition XSD, il convient d'ajouter le wrapper de la propriété de l'objet "$schema" de la définition de la composante utilisée. Les attributs de l'élément de base d'un schéma XML "xsd:schema" n'ont pas d'espaces qualifiés équivalents dans les schémas JSON.

[TR-04] L'élément xsd:schema et ses valeurs d'attribut DEVRAIENT être convertis en propriétés JSON correspondantes à partir des définitions de schéma XML sources comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:schema	"$schema"	
@xmlns	~~ignoré~~S.O.	ignoré
@targetNamespace	~~ignoré~~S.O.	ignoré
@elementFormDefault	~~ignoré~~S.O.	ignoré
@attributeFormDefault	~~ignoré~~S.O.	ignoré
@version	"description"	Concaténé avec la valeur de description avec l'étiquette "Version:". Veuillez vous référer aux règles JSD-10 et TR-20.

Quelques exemples sont proposés ci-dessous :

Définition de la composante du schéma XML (AbstractNumber.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:element name="AbstractNumber" type="xsd:string">
       <xsd:annotation>
           <xsd:documentation>Number assigned to an abstract published without the
full document in a collection of abstracts. This collection can be a journal,
conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
Abstracts), etc.</xsd:documentation>
       </xsd:annotation>
</xsd:element>
```

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

Définition de la composante du schéma JSON (abstractNumber.json)

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

Définition du type de composante du schéma XML (AdditionalRemarkType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="languageCode.xsd"/>
   <xsd:complexType name="AdditionalRemarkType">
      <xsd:sequence>
         <xsd:element ref="com:P"/>
      </xsd:sequence>
      <xsd:attribute ref="com:languageCode"/>
   </xsd:complexType>
</xsd:schema>
```

Définition du type de composante du schéma JSON (additionalRemarkType.json)

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

Définition d'une composante de schéma XML au niveau du document (DesignApplication_V5_0.xsd)

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

    <com:SchemaReleaseNoteURL>http://www.wipo.int/standards/XMLSchema/ST96/V5_0/Release
Notes.pdf</com:SchemaReleaseNoteURL>
        </xsd:appinfo>
    </xsd:annotation>
<xsd:include schemaLocation="DesignApplicationType_V5_0.xsd"/>
    <xsd:element name="DesignApplication" type="dgn:DesignApplicationType">
        <xsd:annotation>
            <xsd:documentation>Details on a design application</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

Définition d'une composante de schéma JSON au niveau du document (designApplication_V5_0.json)

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

Définition d'un type de composante du schéma XML au niveau du document (DesignApplicationType_V5_0.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:dgn="http://www.wipo.int/standards/XMLSchema/ST96/Design"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Design"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RequestSoftware.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationFormName.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RequestExamination.xsd"/>
```

```
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RegistrationOfficeCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReceivingOfficeCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReceivingOfficeDate.xsd"/>
    <xsd:include schemaLocation="../ReceiptNumber.xsd"/>
    <xsd:include schemaLocation="../SealedDepositIndicator.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationNumber.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/FilingPlace.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicantFileReference.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationLanguageCode.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/SecondLanguageCode.xsd"/>
    <xsd:include schemaLocation="../DesignTotalQuantity.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/CorrespondenceLanguageCode.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationCurrentStatusCategory.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationCurrentStatusDate.xsd"/>
    <xsd:include schemaLocation="../DesignatedCountryBag.xsd"/>
    <xsd:include schemaLocation="../DesignBag.xsd"/>
    <xsd:include schemaLocation="../ApplicantBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/RepresentativeBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/Authorization.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationEventBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/DocumentIncludedBag.xsd"/>
    <xsd:include schemaLocation="../DesignApplicationStatementBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/PaymentBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ReimbursementBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/SignatureBag.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationDate.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ApplicationDateTime.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/CorrespondenceAddress.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/BusinessEntityStatusCategory.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/InternationalRegistrationNumber.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/operationCategory.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/st96Version.xsd"/>
    <xsd:import namespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
schemaLocation="../../Common/ipoVersion.xsd"/>
    <xsd:complexType name="DesignApplicationType">
        <xsd:sequence>
            <xsd:element ref="com:RequestSoftware" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationFormName" minOccurs="0"/>
            <xsd:element ref="com:RequestExamination" minOccurs="0"/>
            <xsd:element ref="com:RegistrationOfficeCode"/>
            <xsd:element ref="com:ReceivingOfficeCode" minOccurs="0"/>
            <xsd:element ref="com:ReceivingOfficeDate" minOccurs="0"/>
            <xsd:element ref="dgn:ReceiptNumber" minOccurs="0"/>
```

```
            <xsd:element ref="dgn:SealedDepositIndicator" minOccurs="0"/>
            <xsd:element ref="com:ApplicationNumber" minOccurs="0"/>
            <xsd:element ref="com:InternationalRegistrationNumber" minOccurs="0"/>
            <xsd:element ref="com:FilingPlace" minOccurs="0"/>
            <xsd:element ref="com:ApplicantFileReference" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationLanguageCode" minOccurs="0"/>
            <xsd:element ref="com:SecondLanguageCode" minOccurs="0"/>
            <xsd:element ref="dgn:DesignTotalQuantity" minOccurs="0"/>
            <xsd:element ref="com:CorrespondenceLanguageCode" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationCurrentStatusCategory"
minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationCurrentStatusDate" minOccurs="0"/>
            <xsd:element ref="dgn:DesignatedCountryBag" minOccurs="0"/>
            <xsd:element ref="dgn:DesignBag"/>
            <xsd:element ref="dgn:ApplicantBag"/>
            <xsd:element ref="com:RepresentativeBag" minOccurs="0"/>
            <xsd:element ref="com:Authorization" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationEventBag" minOccurs="0"/>
            <xsd:element ref="com:DocumentIncludedBag" minOccurs="0"/>
            <xsd:element ref="dgn:DesignApplicationStatementBag" minOccurs="0"/>
            <xsd:element ref="com:PaymentBag" minOccurs="0"/>
            <xsd:element ref="com:ReimbursementBag" minOccurs="0"/>
            <xsd:element ref="com:SignatureBag" minOccurs="0"/>
            <xsd:choice>
                <xsd:element ref="com:ApplicationDate" minOccurs="0"/>
                <xsd:element ref="com:ApplicationDateTime"/>
            </xsd:choice>
            <xsd:element ref="com:CorrespondenceAddress" minOccurs="0"/>
            <xsd:element ref="com:BusinessEntityStatusCategory" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute ref="com:operationCategory"/>
        <xsd:attribute ref="com:st96Version" use="required"/>
        <xsd:attribute ref="com:ipoVersion"/>
    </xsd:complexType>
</xsd:schema>
```

Définition d'un type de composante du schéma JSON au niveau du document (designApplicationType_V5_0.json)

```
{
  "$id" : "designApplicationType_V5_0.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$defs" : {
    "designApplicationType" : {
      "description" : "Version: V5_0",
      "type" : "object",
      "additionalProperties" : false,
      "properties" : {
        "operationCategory" : {
          "$ref" : "../../Common/operationCategory.json#/$defs/operationCategory"
        },
        "st96Version" : {
          "$ref" : "../../Common/st96Version.json#/$defs/st96Version"
        },
        "ipoVersion" : {
          "$ref" : "../../Common/ipoVersion.json#/$defs/ipoVersion"
        },
        "requestSoftware" : {
          "$ref" : "../../Common/requestSoftware.json#/$defs/requestSoftware"
        },
        "designApplicationFormName" : {
          "$ref" :
"../designApplicationFormName.json#/$defs/designApplicationFormName"
        },
        "requestExamination" : {
```

```
          "$ref" : "../../Common/requestExamination.json#/$defs/requestExamination"
        },
        "registrationOfficeCode" : {
          "$ref" :
"../../Common/registrationOfficeCode.json#/$defs/registrationOfficeCode"
        },
        "receivingOfficeCode" : {
          "$ref" : "../../Common/receivingOfficeCode.json#/$defs/receivingOfficeCode"
        },
        "receivingOfficeDate" : {
          "$ref" : "../../Common/receivingOfficeDate.json#/$defs/receivingOfficeDate"
        },
        "receiptNumber" : {
          "$ref" : "../receiptNumber.json#/$defs/receiptNumber"
        },
        "sealedDepositIndicator" : {
          "$ref" : "../sealedDepositIndicator.json#/$defs/sealedDepositIndicator"
        },
        "applicationNumber" : {
          "$ref" : "../../Common/applicationNumber.json#/$defs/applicationNumber"
        },
        "internationalRegistrationNumber" : {
          "$ref" :
"../../Common/internationalRegistrationNumber.json#/$defs/internationalRegistrationNum
ber"
        },
        "filingPlace" : {
          "$ref" : "../../Common/filingPlace.json#/$defs/filingPlace"
        },
        "applicantFileReference" : {
          "$ref" :
"../../Common/applicantFileReference.json#/$defs/applicantFileReference"
        },
        "designApplicationLanguageCode" : {
          "$ref" :
"../designApplicationLanguageCode.json#/$defs/designApplicationLanguageCode"
        },
        "secondLanguageCode" : {
          "$ref" : "../../Common/secondLanguageCode.json#/$defs/secondLanguageCode"
        },
        "designTotalQuantity" : {
          "$ref" : "../designTotalQuantity.json#/$defs/designTotalQuantity"
        },
        "correspondenceLanguageCode" : {
          "$ref" :
"../../Common/correspondenceLanguageCode.json#/$defs/correspondenceLanguageCode"
        },
        "designApplicationCurrentStatusCategory" : {
          "$ref" :
"../designApplicationCurrentStatusCategory.json#/$defs/designApplicationCurrentStatusC
ategory"
        },
        "designApplicationCurrentStatusDate" : {
          "$ref" :
"../designApplicationCurrentStatusDate.json#/$defs/designApplicationCurrentStatusDate"
        },
        "designatedCountryBag" : {
          "$ref" : "../designatedCountryBag.json#/$defs/designatedCountryBag"
        },
        "designBag" : {
          "$ref" : "../designBag.json#/$defs/designBag"
        },
        "applicantBag" : {
          "$ref" : "../applicantBag.json#/$defs/applicantBag"
```

```
        },
        "representativeBag" : {
          "$ref" : "../../Common/representativeBag.json#/$defs/representativeBag"
        },
        "authorization" : {
          "$ref" : "../../Common/authorization.json#/$defs/authorization"
        },
        "designApplicationEventBag" : {
          "$ref" :
"../designApplicationEventBag.json#/$defs/designApplicationEventBag"
        },
        "documentIncludedBag" : {
          "$ref" : "../../Common/documentIncludedBag.json#/$defs/documentIncludedBag"
        },
        "designApplicationStatementBag" : {
          "$ref" :
"../designApplicationStatementBag.json#/$defs/designApplicationStatementBag"
        },
        "paymentBag" : {
          "$ref" : "../../Common/paymentBag.json#/$defs/paymentBag"
        },
        "reimbursementBag" : {
          "$ref" : "../../Common/reimbursementBag.json#/$defs/reimbursementBag"
        },
        "signatureBag" : {
          "$ref" : "../../Common/signatureBag.json#/$defs/signatureBag"
        },
        "applicationDate" : {
          "$ref" : "../../Common/applicationDate.json#/$defs/applicationDate"
        },
        "applicationDateTime" : {
          "$ref" : "../../Common/applicationDateTime.json#/$defs/applicationDateTime"
        },
        "correspondenceAddress" : {
          "$ref" :
"../../Common/correspondenceAddress.json#/$defs/correspondenceAddress"
        },
        "businessEntityStatusCategory" : {
          "$ref" :
"../../Common/businessEntityStatusCategory.json#/$defs/businessEntityStatusCategory"
        }
      },
      "oneOf" : [ {
        "required" : [ "applicationDate" ]
      }, {
        "required" : [ "applicationDateTime" ]
      } ],
      "required" : [ "st96Version", "registrationOfficeCode", "designBag",
"applicantBag" ]
    }
  }
}
```

RÉFÉRENCEMENT DE SCHÉMA GÉNÉRAL

[TR-05] Les deux manières conventionnelles de faire référence à des définitions générales utilisées dans les définitions de schéma XML, à savoir `xsd:import` et `xsd:include`, DOIVENT être converties avec la propriété "`$ref`" dans le schéma JSON, quel que soit le type utilisé dans la définition de schéma XML "`import`" ou "`include`".

Par exemple :

Définition de schéma XML pour l'exemple xsd:import (RelatedApplicationDate.xsd)

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

Définition de schéma JSON pour l'exemple xsd:import (relatedApplicationDate.json)

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

Définition de schéma XML pour l'exemple xsd:include (AffectedDesign.xsd)

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

Définition de schéma JSON pour l'exemple xsd:include (affectedDesign.json)

```
{
  "$id" : "affectedDesign.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
```

```
    "affectedDesign" : {
      "$ref" : "#/$defs/affectedDesign"
    }
  },
  "required" : [ "affectedDesign" ],
  "$defs" : {
    "affectedDesign" : {
      "$ref" : "affectedDesignType.json#/$defs/affectedDesignType",
      "description" : "Description: Design affected by the decision, either all
designs or enumeration; Version: V5_0"
    }
  }
}
```

COMPOSITEURS XSD

Les compositeurs sont les structures du schéma W3C qui regroupent les déclarations d'éléments. Il y a trois catégories de compositeurs dans la norme du schéma W3C : `sequence`, `choice` et `all`. L'élément `xsd:all` ne peut pas être utilisé dans la norme ST.96 conformément à la règle [SD-52] de l'annexe I de la norme ST.96 de l'OMPI.

[TR-06] La propriété d'un objet ou d'un tableau d'une définition de la composante utilisée DEVRAIT être ajoutée en fonction de la valeur "`maxOccurs`" comme indiqué ci-dessous.

XSD	JSON	Notes
`xsd:sequence`	objet	
`maxOccurs=1`	objet	Avec la valeur `minOccurs=1`, l'objet est nécessaire
`maxOccurs=unbounded`	objet ou tableau	Avec la valeur `minOccurs=1`, l'objet est nécessaire
`xsd:choice`	objet	
`maxOccurs=1`	objet	Avec la valeur `minOccurs=1`, l'objet est nécessaire
`maxOccurs=unbounded`	objet ou tableau	Avec la valeur `minOccurs=1`, l'objet est nécessaire

Par exemple :

Définition de schéma XML pour l'exemple xsd:sequence maxOccurs=1 (AdditionalRemarkType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="languageCode.xsd"/>
    <xsd:complexType name="AdditionalRemarkType">
        <xsd:sequence>
            <xsd:element ref="com:P"/>
        </xsd:sequence>
        <xsd:attribute ref="com:languageCode"/>
    </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:sequence maxOccurs=1 (additionalRemarkType.json)

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

```
          "$ref" : "languageCode.json#/$defs/languageCode"
        },
        "p" : {
          "$ref" : "p.json#/$defs/p"
        }
      },
      "required" : [ "p" ]
    }
  }
}
```

Définition de schéma XML pour l'exemple xsd:sequence maxOccurs=unbounded (InventionClaimBagType.xsd)

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

Définition de schéma JSON pour l'exemple xsd:sequence maxOccurs=unbounded (inventionClaimBagType.json)

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

```
            "$ref" : "claimNumberRange.json#/$defs/claimNumberRange"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "claimNumberRange.json#/$defs/claimNumberRange"
            }
          } ]
        }
      },
      "anyOf" : [ {
        "required" : [ "claimNumber" ]
      }, {
        "required" : [ "claimNumberRange" ]
      } ],
      "required" : [ "inventionNumber" ]
    }
  }
}
```

Définition de schéma XML pour l'exemple xsd:choice maxOccurs=1 (ChemicalFormulaeType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="Image.xsd"/>
<xsd:include schemaLocation="InlineFormula.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:include schemaLocation="chemicalFormulaeNumber.xsd"/>
<xsd:include schemaLocation="ExternalDocumentBag.xsd"/>
    <xsd:complexType name="ChemicalFormulaeType">
        <xsd:choice>
            <xsd:element ref="com:Image"/>
            <xsd:element ref="com:InlineFormula"/>
            <xsd:element ref="com:ExternalDocumentBag"/>
        </xsd:choice>
        <xsd:attribute ref="com:id"/>
        <xsd:attribute ref="com:chemicalFormulaeNumber"/>
    </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:choice maxOccurs=1 (chemicalFormulaeType.json)

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

```
        "inlineFormula" : {
          "$ref" : "inlineFormula.json#/$defs/inlineFormula"
        },
        "externalDocumentBag" : {
          "$ref" : "externalDocumentBag.json#/$defs/externalDocumentBag"
        }
      },
      "oneOf" : [ {
        "required" : [ "image" ]
      }, {
        "required" : [ "inlineFormula" ]
      }, {
        "required" : [ "externalDocumentBag" ]
      } ]
    }
  }
}
```

Définition de schéma XML pour l'exemple xsd:choice maxOccurs=unbounded (InventionClaimBagType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="Heading.xsd"/>
<xsd:include schemaLocation="P.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:complexType name="ContentType">
       <xsd:choice maxOccurs="unbounded">
<xsd:element ref="com:Heading"/>
<xsd:element ref="com:P"/>
       </xsd:choice>
       <xsd:attribute ref="com:id"/>
   </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:choice maxOccurs=unbounded (inventionClaimBagType.json)

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

```
          "anyOf" : [ {
            "$ref" : "p.json#/$defs/p"
          }, {
            "type" : "array",
            "minItems" : 1,
            "items" : {
              "$ref" : "p.json#/$defs/p"
            }
          } ]
        }
      },
      "anyOf" : [ {
        "required" : [ "heading" ]
      }, {
        "required" : [ "p" ]
      } ]
    }
  }
```

ÉLÉMENTS

Les éléments sont le socle d'un document XML et ils doivent être convertis en propriété dans JSON.

[TR-07] La propriété avec un type approprié DOIT être ajoutée à la définition de la composante utilisée en fonction de la valeur `maxOccurs` comme indiqué ci-dessous.

XSD	JSON	Notes
`xsd:element`	Propriété	Voir la section relative aux types de données incorporées ci-dessus
`maxOccurs=1`	Propriété avec un type XSD primitif ou personnalisé	Avec la valeur `minOccurs=1`, la propriété est nécessaire
`maxOccurs=unbounded`	Tableau de type primitif ou personnalisé	Avec la valeur `minOccurs=1`, la propriété est nécessaire

Par exemple :

Définition de schéma XML pour l'exemple xsd:element maxOccurs=1 (DocumentTotalQuantity.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:element name="DocumentTotalQuantity" type="xsd:nonNegativeInteger">
        <xsd:annotation>
            <xsd:documentation>Total number of documents available or provided.
</xsd:documentation>
        </xsd:annotation>
    </xsd:element>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:element maxOccurs=1 (documentTotalQuantity.json)

```
{
  "$id" : "documentTotalQuantity.json",
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "additionalProperties" : false,
  "properties" : {
    "documentTotalQuantity" : {
      "$ref" : "#/$defs/documentTotalQuantity"
    }
  },
  "required" : [ "documentTotalQuantity" ],
```

```
  "$defs" : {
    "documentTotalQuantity" : {
      "type" : "integer",
      "minimum" : 0,
      "description" : "Description: Total number of documents available or provided.
; Version: V5_0"
    }
  }
}
```

Définition de schéma XML pour l'exemple xsd:element maxOccurs=unbounded (IPOfficeCodeBagType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="IPOfficeCode.xsd"/>
   <xsd:complexType name="IPOfficeCodeBagType">
      <xsd:sequence>
         <xsd:element ref="com:IPOfficeCode" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:element maxOccurs=unbounded (ipOfficeCodeBagType.json)

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

ATTRIBUTS

Les attributs sont les structures du schéma W3C associées à des éléments qui fournissent des informations additionnelles sur les éléments et doivent être convertis en propriété de la composante utilisée dans JSON.

[TR-08] La propriété avec le type approprié DOIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous :

XSD	JSON	Notes
`xsd:attribute`	Propriété ou composante utilisée	Voir la section "Conversion des types de données incorporées"
`xsd:annotation`	"`$description`"	Voir la section "Annotation"

Par exemple :

Définition de schéma XML pour l'exemple xsd:attribute (changeDateTime.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:attribute name="changeDateTime" type="xsd:dateTime">
       <xsd:annotation>
           <xsd:documentation>Date and time of change</xsd:documentation>
       </xsd:annotation>
   </xsd:attribute>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:attribute (changeDateTime.json)

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

SIMPLETYPE (TYPE SIMPLE)

[TR-09] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:simpleType	objet	Voir la section "Conversion des types de données incorporées"
xsd:union	objet – "anyOf"	Voir la section "Union"
xsd:restriction	objet	Voir la section "Restriction"
xsd:enumeration	"enum"	Voir la section "Énumération"

COMPLEXTYPE (TYPE COMPLEXE)

[TR-10] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
Xsd:complexType	objet	Voir la section "Conversion des types de données incorporées"
xsd:simpleContent	objet	Voir la section "Contenu simple"
xsd:complexContent	objet	Voir la section "Contenu complexe"
xsd:sequence	objet	Voir la section "Compositeurs XSD"
xsd:choice	objet/tableau	Voir la section "Compositeurs XSD"
@mixed	objet	Voir la section "Contenu mixte"

Contenu simple

[TR-11] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:simpleContent	objet	

xsd:extension	objet	Voir la section “Extension”
xsd:restriction	objet	Voir la section “Restriction”

Contenu complexe

[TR-12] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:complexContent	objet	
xsd:extension	objet	Voir la section “Extension”
xsd:restriction	objet	Voir la section “Restriction”

Contenu mixte

[TR-13] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:complexType	objet	
@mixed	objet	

Par exemple :

Définition de schéma XML pour l’exemple xsd:complexType\@mixed=true (CrossReferenceType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="PhraseType.xsd"/>
<xsd:include schemaLocation="id.xsd"/>
<xsd:include schemaLocation="idrefs.xsd"/>
<xsd:include schemaLocation="extRef.xsd"/>
<xsd:include schemaLocation="crossReferenceCategory.xsd"/>
<xsd:include schemaLocation="sourceURI.xsd"/>
<xsd:include schemaLocation="sourceSystemName.xsd"/>
<xsd:include schemaLocation="sourceSystemIdentifier.xsd"/>
    <xsd:complexType name="CrossReferenceType" mixed="true">
        <xsd:complexContent>
            <xsd:extension base="com:PhraseType">
                <xsd:attribute ref="com:id"/>
                <xsd:attribute ref="com:idrefs"/>
                <xsd:attribute ref="com:extRef"/>
                <xsd:attribute ref="com:crossReferenceCategory" use="required"/>
                <xsd:attribute ref="com:sourceURI"/>
                <xsd:attribute ref="com:sourceSystemName"/>
                <xsd:attribute ref="com:sourceSystemIdentifier"/>
                </xsd:extension>
         </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour l’exemple xsd:complexType\@mixed=true (crossReferenceType.xsd)

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

```
        },
        "id" : {
          "$ref" : "id.json#/$defs/id"
        },
        "idrefs" : {
          "$ref" : "idrefs.json#/$defs/idrefs"
        },
        "extRef" : {
          "$ref" : "extRef.json#/$defs/extRef"
        },
        "crossReferenceCategory" : {
          "$ref" : "crossReferenceCategory.json#/$defs/crossReferenceCategory"
        },
        "sourceURI" : {
          "$ref" : "sourceURI.json#/$defs/sourceURI"
        },
        "sourceSystemName" : {
          "$ref" : "sourceSystemName.json#/$defs/sourceSystemName"
        },
        "sourceSystemIdentifier" : {
          "$ref" : "sourceSystemIdentifier.json#/$defs/sourceSystemIdentifier"
        }
      },
      "required" : [ "crossReferenceCategory" ]
    }
  }
}
```

ANNOTATION

Deux types d'annotation sont utilisés dans la norme ST.96 de l'OMPI, à savoir `xsd:appinfo` et `xsd:documentation`.

`xsd:appinfo`

L'élément `xsd:appinfo` précise la nature de l'information utilisée par l'application. Il doit être placé dans un élément d'annotation. La norme ST.96 utilise l'élément `xsd:appinfo` dans toutes les définitions de schéma XML au niveau du document contenant les éléments de schéma ci-dessous :

- `com:SchemaCreatedDate`
- `com:SchemaLastModifiedDate`
- `com:SchemaContactPoint`
- `com:SchemaReleaseNoteURL`

[TR-14] L'information fournie sous la balise `<xsd:appinfo>` DEVRAIT être déplacée à la valeur "description" dans un schéma JSON.

Par exemple :

Définition de la composante d'un schéma XML (ApplicationBodyType.xsd)

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

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

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

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

Définition de schéma JSON pour l'exemple xsd:annotation (applicationBodyType.json)

```
"$defs" : {
      "applicationBody" : {
            "$ref" : "applicationBodyType V5 0.json#/$defs/applicationBodyType",
                "description" : "Description: Body of a patent application; Version:
      V5_0; SchemaCreatedDate: 2012-07-13; SchemaLastModifiedDate: 2021-10-01;
      SchemaContactPoint: xml.standards@wipo.int;
      SchemaReleaseNoteURL: http://www.wipo.int/standards/XMLSchema/ST96/V5_0/ReleaseNo
      tes.pdf"
             }

      }
```

xsd:documentation

Dans la norme ST.96, cet élément précise l'information qui doit être lue ou utilisée par les utilisateurs dans un élément d'annotation. Selon la norme ST.96 :

> "*[SD-58], Tous les schémas DEVRAIENT inclure une documentation sur leurs structures utilisant l'élément xsd:documentation."*

[TR-15] Les informations relatives à la documentation et à la version DEVRAIENT être déplacées à la valeur "description" dans le schéma JSON.

Par exemple :

Définition de la composante du schéma XML (AbstractNumber.xsd)

```
    <xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
    elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
      <xsd:element name="AbstractNumber" type="xsd:string">
            <xsd:annotation>
                  <xsd:documentation>Number assigned to an abstract published without
    the full document in a collection of abstracts. This collection can be a journal,
    conference proceedings, a patent collection of abstracts (e.g. Soviet Patent
    Abstracts), etc.</xsd:documentation>
            </xsd:annotation>
      </xsd:element>
    </xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:documentation (abstractNumber.json) :

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

UNION

[TR-16] Une propriété avec un type approprié DEVRAIT être ajoutée à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
xsd:union	objet – "anyOf"	Voir la section "Conversion des types de données incorporées"

Par exemple:

Définition de schéma XML pour l'exemple xsd:simpleType\xsd:union (DocumentNameType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="DocumentNameCategoryType.xsd"/>
<xsd:simpleType name="DocumentNameType">
    <xsd:union memberTypes="xsd:string com:DocumentNameCategoryType"/>
    </xsd:simpleType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:simpleType\xsd:union (documentNameType.json)

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

EXTENSION

[TR-17] Un objet DEVRAIT être ajouté à la définition de la composante utilisée comme indiqué ci-dessous :

XSD	JSON	Notes
xsd:extension	objet – "anyOf"	Voir la section "Conversion des types de données incorporées"

Par exemple:

Définition de schéma XML pour l'exemple xsd:extension (AmountType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:include schemaLocation="currencyCode.xsd"/>
<xsd:complexType name="AmountType">
       <xsd:simpleContent>
           <xsd:extension base="xsd:decimal">
               <xsd:attribute ref="com:currencyCode"/>
           </xsd:extension>
       </xsd:simpleContent>
   </xsd:complexType>
</xsd:schema>
```

Définition de schéma JSON pour xsd:extension (amountType.json)

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

RESTRICTION

[TR-18] Un objet DEVRAIT être ajouté à la définition de la composante utilisée comme indiqué ci-dessous.

XSD	JSON	Notes
`xsd:restriction`	objet	
`xsd:pattern`	style	Voir la section “Style”
`xsd:enumeration`	“*enum*”	Voir la section “Énumération”

ÉNUMÉRATION

[TR-19] L’élément `xsd:enumeration` DOIT être converti en propriété “`enum`” dans la définition de la composante utilisée du schéma JSON.

Définition de schéma XML pour l’exemple xsd:simpleType\xsd:restriction\xsd:enumeration (BusinessEntityStatusCategoryType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
    <xsd:simpleType name="BusinessEntityStatusCategoryType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="Undiscounted">
                <xsd:annotation>
                    <xsd:documentation>Undiscounted entity</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="Small">
                <xsd:annotation>
                    <xsd:documentation>Small entity discount</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="Micro">
                <xsd:annotation>
                    <xsd:documentation>Micro entity discount</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:simpleType\xsd:restriction\xsd:enumeration (businessEntityStatusCategoryType.json)

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

FACETTES CONTRAIGNANTES

Les schémas W3C utilisent les facettes contraignantes énumérées dans le tableau ci-dessous. Il convient de noter que les éléments `xsd:minInclusive`, `xsd:maxInclusive`, `xsd:minExclusive`, `xsd:maxExclusive`, et `xsd:minLength` ne sont pas utilisés dans la norme ST.96. Par conséquent, ces facettes non utilisées ne figurent pas à l'annexe I.

[TR-20] Les facettes contraignantes XSD DOIVENT être converties en mots clés JSON correspondants accompagnés de la longueur minimale ou maximale appropriée comme indiqué dans le tableau ci-dessous.

Tableau 3 : Conversion des facettes contraignantes XSD (X est la valeur numérique limitée)

Facette contraignante XSD	Équivalent dans le schéma JSON
`<xsd:minLength value="X" />`	`{ "minLength": X }`
`<xsd:maxLength value="X" />`	`{ "maxLength": X }`
`<xsd:length value="X" />`	`{ "minLength": X, "maxLength": X }`
`<xsd:pattern value="X" />`	`{ "pattern": "X" }`
`<xsd:minExclusive value="X" />`	`{ "minimum": X, "exclusiveMinimum": true }`
`<xsd:maxExclusive value="X" />`	`{ "maximum": X, "exclusiveMaxmimum": true }`
`<xsd:minInclusive value="X" />`	`{ "minimum": X, "exclusiveMinimum": false }`
`<xsd:maxInclusive value="X" />`	`{ "maximum": X, "exclusiveMaximum": false }`

Par exemple :

Définition de schéma XML pour l'exemple xsd:length (ClassType.xsd)

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

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

Définition de schéma JSON pour l'exemple xsd:length (classType.json)

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

Style

[TR-21] L'élément `xsd:pattern` DOIT être converti en une propriété "`pattern`" dans la définition de la composante utilisée.

Par exemple :

Définition de schéma XML pour l'exemple xsd:pattern (WIPONotificationNumberType.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:com="http://www.wipo.int/standards/XMLSchema/ST96/Common"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.wipo.int/standards/XMLSchema/ST96/Common"
elementFormDefault="qualified" attributeFormDefault="qualified" version="V5_0">
<xsd:simpleType name="WIPONotificationNumberType">
       <xsd:restriction base="xsd:token">
           <xsd:pattern value="[A-Z]{3}[0-9]{6}"/>
       </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

Définition de schéma JSON pour l'exemple xsd:pattern (wipoNotificationNumberType.json)

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

GROUPE

L'élément `xsd:group` n'est pas utilisé dans la norme ST.96, mais il est utilisé dans les normes externes mentionnées dans les définitions de schéma XML de la norme ST.96. De plus amples informations sur les normes externes sont disponibles dans la section "Conversion des dépendances XSD externes" ci-après.

[TR-22] Le type d'objet DEVRAIT être ajouté à la définition de la composante utilisée.

XSD	JSON	Notes
`xsd:group`	objet	

Par exemple :

Définition de schéma XML pour l'exemple xsd:group (fragment de FlattenedMathML3.json)

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

Définition de schéma JSON pour l'exemple xsd:group (fragment de flattenedMathML3.json)

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

CONVERSION DES DÉPENDANCES XSD EXTERNES

La norme ST.96 de l'OMPI renvoie aux schémas des normes industrielles ci-après, au lieu de les redéfinir dans la norme ST.96 :

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

[3] http://www.w3.org/TR/MathML3

– Schéma de tableau OASIS[4] (`OASISTable_V1_0.xsd`).

Au moment de l'élaboration de la présente norme, ces normes externes ne fournissent pas de schéma JSON équivalent. Cependant, reconnaissant que les schémas JSON fondés sur la norme ST.96 ne sont pas possibles sans schémas JSON équivalents, comme solution programmatique, ces deux XSD externes sont convertis en schémas JSON équivalents relativement stables à l'aide de l'outil de conversion figurant dans l'appendice de l'annexe I de la présente norme et les schémas JSON convertis sont inclus dans l'ensemble de schémas JSON qui fait l'objet de l'annexe II.

[L'appendice de l'annexe I de la proposition de norme suit]

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

APPENDICE

OUTIL DE CONVERSION DE DÉFINITION DE SCHÉMA XML EN SCHÉMA JSON

Le présent appendice de l'annexe I contient l'outil de conversion des schémas XML en schémas JSON, qui est une bibliothèque Java qui aide les offices de propriété intellectuelle à convertir une définition de schéma XML fondée sur la norme ST.96 de l'OMPI en son schéma JSON équivalent, conformément aux règles de conversion figurant à l'annexe I. Cet outil de conversion est fourni ici dans le cadre de la norme afin que les offices de propriété intellectuelle puissent également utiliser cet outil pour convertir leurs propres définitions de schéma XML de la norme ST.96 personnalisées en schéma JSON.

Exigences

Java Runtime Environment 1.8 (ou version supérieure)

Usage

~~C:\>~~#Fournit des options d'aide
```
C:\>java -jar xsd2JsonSchema.jar -help
```

~~C:\>~~#Transformer un fichier (XSD)
```
C:\>java -jar xsd2JsonSchema.jar -f "C:\XSD_Folder_Path\Common\AbstractNumber.xsd"
```

~~C:\>~~#Transformer un fichier (XSD) et les schémas inclus/importés
```
C:\>java -jar xsd2JsonSchema.jar -f -r "C:\XSD_Folder_Path\Common\AbstractNumber.xsd"
```

~~C:\>~~#Transformer tous les schémas du répertoire fourni (l'indicateur récursif n'est pas disponible avec cette option)
```
C:\>java -jar xsd2JsonSchema.jar -d "C:\XSD_Folder_Path\Common"
```

Télécharger le fichier exécutable Jar

L'outil peut être téléchargé dans l'appendice de l'annexe I en cliquant sur le lien suivant : https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixi.zip.

[L'annexe II de la proposition de norme suit]

ANNEXE II

SCHÉMA JSON

[Notes : Les règles de conversion figurant à l'annexe I s'appliquent également aux normes XML externes converties, c'est-à-dire MathML et tableau OASIS. Ces noms de balise ont été conservés en l'état tout comme les noms des définitions de schéma XML de la norme ST.96, mais la règle des caractères minuscules de type "camel" (LCC) s'applique à tous les noms, p. ex, MathExpression à mathExpression dans MathML, ce qui n'est pas prévu. Tous les noms de balise des normes XML externes devraient être conservés en l'état. Par conséquent, les noms de balise modifiés seront rétablis pour la publication de la nouvelle norme.]

L'annexe II fournit un ensemble complet de schémas JSON qui correspond aux définitions de schéma XML de la version 5.0. de la norme ST.96 de l'OMPI. Ces schémas JSON ont été générés automatiquement à l'aide de l'outil de conversion fourni dans l'appendice de l'annexe I, conformément aux règles et aux principes de conversion définis à l'annexe I. Il convient de noter qu'il s'agit d'un processus de conversion à sens unique, c'est-à-dire de XSD en JSON. Cet ensemble de schémas JSON comprend les schémas JSON convertis des normes XML externes, c'est-à-dire MathML et tableau OASIS. Toutefois, les noms de balise originaux des composantes des définitions de schéma XML des normes XML externes sont conservés en l'état.

Les schémas JSON peuvent être téléchargés dans l'appendice de l'annexe II en cliquant sur le lien suivant : https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixii.zip.

[L'annexe III de la proposition de norme suit]

ANNEXE III

EXEMPLES D'INSTANCES JSON

L'annexe III fournit des exemples d'instances JSON établis à partir des définitions de schéma XML de la norme ST.96 de l'OMPI afin d'aider les offices de propriété intellectuelle à produire des instances similaires. Chacune de ces instances devrait être validée en fonction du schéma JSON pertinent qui figure à l'annexe II.

Les exemples d'instances ci-après correspondent à certaines composantes au niveau du document, qui figurent à l'annexe VII de la norme ST.96 de l'OMPI, et ils ne représentent pas des données réelles :

- **`patentPublication.json`** : cette composante au niveau du document est utilisée pour saisir les détails de la publication d'une demande de brevet. Cet exemple d'instance peut être téléchargé ici : *patentPublication.json*
- **`trademarkApplication.json`** : cette composante au niveau du document est utilisée pour saisir les informations relatives à un dépôt de demande de marque. Cet exemple d'instance peut être téléchargé ici : trademarkApplication.json
- **`designApplication.json`** : cette composante au niveau du document est utilisée pour saisir les informations relatives à un dépôt de demande de dessin ou modèle industriel. Cet exemple d'instance peut être téléchargé ici : designApplication.json

[Note : les liens téléchargeables pour ces exemples d'instances seront fournis ci-dessous après la publication de la proposition de norme actuelle]

Les exemples d'instances JSON peuvent être téléchargés dans l'appendice de l'annexe III en cliquant sur le lien suivant : https://www.wipo.int/edocs/mdocs/cws/en/cws_10/cws_10_6-appendixiii.zip.

[L'annexe IV de la proposition de norme suit]

ANNEXE IV

LISTE DES SIGLES ET ABRÉVIATIONS

Les sigles et abréviations au début d'un type d'objet et d'un nom de propriété DOIVENT figurer en minuscules, par exemple "`pre`", "`bioDeposit`". Si un sigle figure en majuscules au début d'un nom, tous les caractères doivent être en minuscules, par exemple "`idref`" et "`wipo`" dans le nom de propriété de "`wipoST3Code`". Par ailleurs, toutes les valeurs d'une énumération, les valeurs de sigles et les valeurs d'abréviations DOIVENT figurer en majuscules comme indiqué ci-après.

Les sigles et abréviations présentés ci-après NE DEVRAIENT PAS être considérés dans le contexte des codes de langue, de monnaie, d'office ou de pays, qui sont énumérés dans la norme ST.96 de l'OMPI, car ils pourraient produire des doublons. Ces codes sont fondés sur les codes de langue de la norme ISO 639-1, les codes de monnaie de la norme ISO 4217, les codes de la norme ST.3 de l'OMPI et les codes de pays de la norme ISO 3166-1, respectivement.

AF	Fichier d'autorité
Alt	Texte alternatif pour une image
B	Caractère gras
BioDeposit	Dépôt biologique
Br	Saut de ligne
CDX	Format de fichier CambridgeSoft proprietary ChemDraw
CPC	Classification coopérative des brevets
DD	Description de la définition
Del	Texte supprimé
DL	Liste des définitions
DOI	Identificateur d'objet numérique
DT	Terme de définition
DTD	Définition de type de document
DWF	Format de conception sur le Web
DWG	Dessin
ECLA	Classification européenne
EIDR	Registre d'identification des loisirs
ExtRef	Renvois externes au document XML concerné
GI	Indication géographique
H<n>	"n" indique le niveau de l'en-tête avec une valeur spécifique constituée d'un nombre entre 1 et 15. Cela signifie que, dans la valeur d'énumération, cette abréviation indique une valeur entre H1 et H15. Par exemple, "H1" signifie "En tête 1".
I	Italique
IB	Bureau international
ID	Identificateur pour l'identification des systèmes
IDREF	Renvoi à l'identificateur
IDREFS	Renvois à l'identificateur
IGES	Initial Graphic Exchange Specification
IGO	Organisation intergouvernementale dont la constitution prévoit la protection des droits de propriété intellectuelle ou qui agit dans le cadre d'un traité relatif à la propriété intellectuelle
INID	Codes d'identification numérique internationalement agréés en matière de données bibliographiques
Ins	Texte inséré
IP	Propriété intellectuelle
IPC	Classification internationale des brevets
IPCR	Réforme de la classification internationale des brevets
IPO	Office de la propriété intellectuelle
IPR	Droit de propriété intellectuelle
ISMN	Numéro international de la musique
ISNI	Code international normalisé des noms
ISO	Organisation internationale de normalisation
JSON	Javascript Object Notation
LCC	Caractères bas de casse de type "camel"
LI	Liste
LOR	Licence de droit
MathML	Description Mathematical Markup Language
MPEG	Moving Picture Experts Group

MOL	Format de fichier pour disposer d'informations sur les atomes, les liens, la connectivité et les coordonnées d'une molécule
NB	Format de fichier pour les blocs-notes Mathematica
NPL	Littérature non-brevet
NUTS	Nomenclature des unités territoriales statistiques
O	Over score
OASIS	Organization for the Advancement of Structured Information Standards
OCR	Reconnaissance optique des caractères
OL	Liste par ordre
P	Paragraphe
PAN	Numéro de compte principal
PCT	Traité de coopération en matière de brevets
PKCS7	En cryptographie, PKCS est un ensemble de normes de cryptographie à clé publique et PKCS #7 (PKCS7) désigne la norme de syntaxe de message cryptographique qui décrit la syntaxe générale pour les données auxquelles la cryptographie peut s'appliquer, comme les signatures et les enveloppes numériques.
Pre	Texte préformaté
S	Texte biffé
SEQL	Listage de séquence
SOC	Code de société
SPC	Certificat complémentaire de protection
ST3	Norme ST.3 de l'OMPI
ST13	Norme ST.13 de l'OMPI
Sub	Indice
Sup	Exposant
SVG	Image au format Scalable Vector Graphics
SWF	Small Web Format
SWIFT	Society for Worldwide Interbank Financial Telecommunication
ThreeDM	Modélisation en trois dimensions
ThreeDS	3D Studio
TISA	Code alphanumérique du système d'information sur les territoires de la CISAC
TISN	Code numérique du système d'information sur les territoires de la CISAC
TSG	Spécialités traditionnelles garanties
U	Soulignement
UCC	Caractères haut de casse de type "camel"
UL	Liste sans ordre
UPOV	Union internationale pour la protection des obtentions végétales
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WIPO	Organisation Mondiale de la Propriété Intellectuelle
WMV	Windows Media Video

[L'annexe V de la proposition de norme suit]

ANNEXE V

TERMES DE REPRÉSENTATION

Terme	Définition	Type de donnée
Amount (Montant)	Une valeur monétaire.	`number`
Category (Catégorie)	Une division ou un sous-ensemble spécifiquement défini d'un système de classification dans lequel tous les éléments partagent le même concept de taxonomie.	`string`
Code	Une combinaison d'un ou plusieurs chiffres, lettres ou caractères spéciaux, qui est utilisée pour une signification spéciale. Représente des valeurs limites, déterminées à l'avance, ou en format libre.	`string`
Date	La notion d'un moment spécifique, désignée par l'année, le mois et le jour.	`string`, avec le mot clé `"format": "date"`
Identifier (Identificateur)	Une combinaison d'un ou plusieurs nombres entiers, lettres, caractères spéciaux qui identifie de manière unique une instance spécifique d'un objet commercial mais qui peut ne pas avoir une signification facile à définir.	`string`
Indicator (Indicateur)	Un signal de la présence, de l'absence ou de l'exigence de quelque chose. Les valeurs booléennes sont `true` ou `false` (sans guillemets). Ces valeurs sont sensibles à la casse.	`boolean or string`
Measure (Mesure)	Une mesure est une valeur numérique déterminée en mesurant un objet avec l'unité de mesure donnée. `MeasureType` est utilisé pour représenter un type de dimension physique comme la température, la longueur, la vitesse, la largeur, le poids, le volume et la latitude d'un objet. En termes plus précis, `MeasureType` devrait être utilisé pour mesurer les propriétés intrinsèques ou physiques d'un objet considéré comme un tout.	`number`
Name (Nom)	La désignation d'un objet exprimé en un mot ou en une phrase.	`String`
Number (Nombre)	Une série de chiffres ou de caractères alphanumériques désignant l'étiquette, la valeur, la quantité ou l'identification.	`integer`, `number`, or `string`
Percent (Pour cent)	Un chiffre qui représente la partie d'un tout qui sera divisé par 100.	`number`
Quantity (Quantité)	Une quantité est un nombre compté d'unités non monétaires, y compris éventuellement des fractions. `Quantity` est utilisé pour représenter un nombre compté de choses. `Quantity` devrait être utilisé pour des propriétés simples d'un objet considéré comme un composite, une collection ou un conteneur afin de quantifier ou de compter ses éléments. `Quantity` devrait toujours exprimer un nombre compté de choses et la propriété sera totale, expédiée, chargée, stockée. `QuantityType` devrait être utilisé pour les composantes nécessitant des informations sur les unités; et `Integer` devrait être utilisé pour les composantes comptables qui ne nécessitent pas d'informations sur les unités.	`number` avec le mot clé `"minimum": 0` `or integer`
Rate (Taux)	Une quantité ou un montant mesuré en fonction d'une autre quantité ou d'un autre montant.	`number`
Text (Texte)	Une chaîne de caractères formatés ou non formatés, généralement sous la forme de mots (comprend les abréviations et commentaires).	`string`
Time (Temps)	La désignation d'un moment chronologique précis dans une période.	`string`, avec le mot clé `"format": "time"`
DateTime (Date et heure)	La date et l'heure d'un événement.	`string`, avec le mot clé `"format": "date-time"`
URI	L'Uniform Resource Identifier qui identifie l'endroit où se trouve le fichier.	`string`, avec le mot clé `"format":` et les valeurs `"uri" or "uri-reference"`

[Fin de l'annexe V de la proposition de norme et fin de la proposition de norme]

[Fin de l'annexe et du document]