OMPI
ORGANISATION MONDIALE
DE LA PROPRIÉTÉ
INTELLECTUELLE

WIPO Sequence Validator – Manuel d'utilisation

Version 1.1.0

Le présent document a pour but d'aider les offices de propriété intellectuelle à déployer le microservice du logiciel WIPO Sequence Validator et à configurer ce logiciel.

Table des matières

1. Introduction 3
1.1. Aperçu du processus du logiciel 3
1.2. Définition de la structure du système de fichiers du logiciel 5
2. Déploiement du logiciel 6
2.1. Lancement du logiciel par un fichier JAR créé sous Spring Boot 6
2.1.1. Déploiement du logiciel en tant qu'application exécutable 7
2.2. Déploiement en tant que service Web WAR 8
2.2.1. Rapport de vérification 9
2.2.2. Demande du point d'extrémité de rappel 9
2.3. Configuration 16
2.3.1. Paramétrage par défaut 16
2.3.2. Messages en langue locale 18
2.3.3. Noms d'organismes personnalisés 19
2.3.4. Fichiers DTD du format ST.26 servant de référence 19
3. API REST du logiciel 21
3.1. Valider un fichier au format ST.26 de l'OMPI 21
3.2. Statut de validation de la demande 24
Annexe I : Exemple de rapport de vérification 27
Annexe II : Spécification complète de l'API (YAML) 28
Annexe III : Noms des propriétés (JSON) 32

1. Introduction

Le logiciel WIPO Sequence Validator (ci-après appelé “le logiciel”) a pour but de fournir aux offices de propriété intellectuelle un microservice permettant de valider des fichiers XML au format ST.26 de l’OMPI pour s’assurer qu’ils sont conformes à la norme ST.26 de l’OMPI. Tout listage des séquences établi au moyen du logiciel WIPO Sequence est conforme à cette norme; cependant, les utilisateurs peuvent employer d’autres logiciels s’ils le jugent préférable.

Le présent document a pour but d’expliquer la structure, le déploiement et le paramétrage du logiciel, ainsi que le système de fichiers acceptés en entrée; ces éléments sont présentés en détail dans les sections suivantes.

1.1. Aperçu du processus du logiciel

Le logiciel peut être employé dans les quatre cas d’utilisation suivants :

- Valider un fichier au format ST.26 de l’OMPI;
- Demander le statut d’une validation en cours d’exécution;
- Mettre à jour des fichiers de configuration (réservé à l’administrateur de l’office de propriété intellectuelle); et
- Appeler un point d’extrémité de rappel pour lui envoyer le résultat de la validation à la fin du processus. Note : ce point d’extrémité de rappel[1] ne fait pas partie du domaine d’application du logiciel. C’est l’office de propriété intellectuelle chargé d’élaborer et de paramétrer ce service qui doit établir le point d’extrémité.

Le logiciel se compose d’un fichier JAR pouvant être exécuté comme un service Web, ou d’un fichier WAR pouvant être déployé sur un serveur Tomcat.

Dans les deux cas, pour valider un listage des séquences au format ST.26 de l’OMPI, le logiciel prend en entrée des fichiers situés dans un système de fichiers local et génère un rapport de vérification contenant les résultats de la validation; il peut aussi, à titre facultatif, renvoyer les résultats du processus de validation, c’est-à-dire le rapport de vérification, en appelant un point d’extrémité de rappel.

Le principal processus effectué par le logiciel est le suivant :

a) Le système informatique de l’office de propriété intellectuelle concerné enregistre le fichier XML au format ST.26 de l’OMPI dans un dossier par défaut appelé “Inbox”, ou dans le dossier spécifié dans la demande.

b) Il lance ensuite une demande de type HTTP Post pour obtenir la validation du fichier. Selon la configuration, il peut demander une validation complète (“full”) ou de formalité (“formality”). Le processus de validation “de formalité” vérifie que le fichier ST.26 est bien un fichier XML et le valide par rapport à la DTD du format ST.26. Le processus “complet” valide le fichier ST.26 par rapport aux règles de vérification officielles, qui

[1] Dans le présent contexte, un point d’extrémité de rappel est une adresse unique qui est identifiée par une URI et à laquelle des messages de demandes peuvent être envoyés.

sont déduites de la norme ST.26; il effectue aussi le processus de validation “de formalité”.

Note : il est recommandé de réserver le processus de validation “de formalité” au système d’acceptation des dépôts de demande en ligne, car ce processus peut s’exécuter de manière synchrone, tandis que le processus de validation “complet”, qui prend beaucoup plus de temps, est recommandé pour le traitement des demandes par lots.

c) Une fois la validation achevée, la réponse envoyée indique si le fichier a obtenu la validation “de formalité” et, dans le cas où le système informatique de l’office a demandé une validation “complète”, si le processus de validation des règles officielles a été correctement lancé.

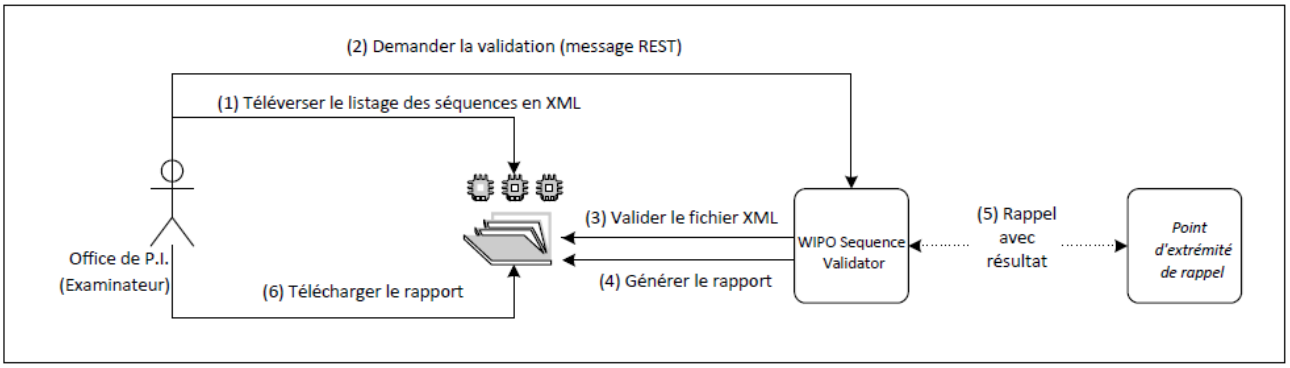
d) Si le logiciel effectue une validation “complète”, il accède au fichier XML dans le dossier “Inbox” et lance le processus de validation selon les règles officielles, puis il exécute les tâches suivantes :

e) Il génère un fichier XML contenant un rapport de vérification (“Verification Report”) dans le dossier “Output” spécifié, et il déplace dans un dossier “Outbox” le fichier XML ST.26 qu’il a validé.

f) Une fois le processus de validation selon les règles officielles achevé, le logiciel appelle le point d’extrémité de rappel, s’il est configuré, en intégrant dans la demande des informations supplémentaires concernant le processus de validation. La structure de la demande et quelques exemples de données sont présentés dans la section 2.2.2 ci-après.

g) Le point d’extrémité de rappel doit répondre soit par un code vide, soit par un code de succès (aucune erreur). [Note : cette étape n’est effectuée que si le service Web extérieur a été rendu accessible et que l’appel a été configuré dans le logiciel.] Il est en outre nécessaire de disposer d’une connectivité entre le logiciel et le point d’extrémité de rappel. Comme nous l’avons indiqué plus haut, le service Web extérieur ne fait pas partie du logiciel et doit être développé et configuré par les offices conformément à la convention définie plus loin.

h) Le système de l’office de propriété intellectuelle peut accéder au rapport de vérification dans le dossier “Report”.



Note : le logiciel WIPO Sequence Validator est conforme à la norme de l'OMPI relative au traitement et à la communication des données de propriété intellectuelle aux API Web : norme ST.90 de l'OMPI.

1.2. Définition de la structure du système de fichiers du logiciel

La structure du système de fichiers employée par le logiciel comporte cinq dossiers :

- **Dossier "Inbox" (Boîte d'entrée)** : Dossier local dans lequel les fichiers ST.26 sont déposés par un office de propriété intellectuelle pour validation.
- **Dossier "Process" (Traitement)** : Dossier local par lequel les fichiers issus du dossier "Inbox" transitent temporairement pendant le traitement. Ce dossier contient deux sous-dossiers :
  - **Dossier "Full validation" (Validation complète)** : Stocke les fichiers en attendant leur validation complète.
  - **Dossier "Formality validation" (Validation de formalité)** : Stocke les fichiers en attendant leur validation de formalité.
- **Dossier "Outbox" (Boîte de sortie)** : Une fois la validation achevée, le logiciel stocke la source du fichier ST.26 dans ce dossier local.
- **Dossier "Report" (Rapport)** : Ce dossier local stocke les résultats de la validation, qui sont enregistrés dans un fichier contenant un rapport de vérification.
- **Dossier "Params" (Paramètres)** : Dossier local dans lequel se trouve un fichier JSON (.json) contenant tous les paramètres de validation définis dans la demande de validation, afin que ces paramètres puissent être utilisés par le processus de validation profonde asynchrone.

On trouvera ci-dessous un exemple de structure de ce système de fichiers :

```
/temp/ST26
/temp/ST26/inbox
/temp/ST26/process/full
/temp/ST26/process/formality
/temp/ST26/outbox
/temp/ST26/reports
/temp/ST26/params
```

2. Déploiement du logiciel

Comme indiqué plus haut, le logiciel est fourni sous deux formats binaires possibles, qui sont indiqués ci-dessous. L'office choisira le format qui lui convient le mieux selon le type d'infrastructure sur laquelle il entend déployer le logiciel.

Les deux formats binaires sous lesquels le logiciel est disponible sont les suivants :

- **Un fichier binaire JAR créé sous Spring Boot** : Ce fichier binaire est un JAR exécutable. Il nécessite l'installation de Java 8.

- **Un paquetage binaire WAR** : Ce fichier binaire est destiné à être déployé sur un conteneur de servlets. Il faut disposer d'un serveur d'applications compatible avec Spring Boot 2 et Servlet Spec 3.1+, comme par exemple Tomcat 8.5.

Les sections ci-après contiennent des indications détaillées sur la manière de déployer le logiciel, que ce soit par une application créée sous Spring Boot ou par un fichier WAR à déployer dans un serveur d'applications Java.

2.1. Lancement du logiciel par un fichier JAR créé sous Spring Boot

Le fichier JAR créé sous Spring Boot contient un serveur intégré qui permet de déployer l'API du logiciel sans avoir besoin d'un serveur indépendant. Ce système simplifie considérablement la configuration et le déploiement en termes d'infrastructures.

Pour lancer le serveur intégré, il faut exécuter la commande suivante :

Note : Java 8 doit être déjà installé sur le serveur : comme Java ne garantit pas l'emploi du codage UTF-8, il faut indiquer "UTF-8" dans le fichier de propriétés système "file.encoding". Ce paramétrage peut être effectué de la manière suivante :

```
java -D"file.encoding=UTF-8" -jar wipo-sequence-validator.jar
```

On peut accéder à l'API du logiciel au moyen de l'application Swagger UI :

http://[host-name:]:8080/swagger-ui.html

On peut accéder à l'API du logiciel depuis les points d'extrémité suivants :

http://[host-name:]:8080/api/[version]/status

http://[host-name:]:8080/api/[version]/validate

L'office de propriété intellectuelle doit apporter les modifications suivantes à cette adresse :

- [host-name] doit être remplacé par le nom du serveur hôte; et

- [version] doit être remplacé par le numéro de version de l'API du logiciel (p. ex. v0.1).

Par défaut, le serveur se lance sur le port 8080; pour modifier le numéro de port, il faut ajouter l'option "--server.port" en ligne de commande, de la manière suivante :

```
java -D"file.encoding=UTF-8" -jar wipo-sequence-validator.jar –-server.port=<port-number>
```

Par défaut, le logiciel utilise les paramètres mémoire de la machine virtuelle de Java (JVM) définis par défaut. La taille du tas (“heap size”) maximale par défaut est un quart de la mémoire physique disponible.

Pour modifier la taille du tas maximale, il faut utiliser l’option “-Xmx” lorsque l’exécution est lancée en ligne de commande[2] :

```
java -D"file.encoding=UTF-8" -Xmx[size]-jar wipo-sequence-validator.jar
```

2.1.1. Déploiement du logiciel en tant qu’application exécutable

Le logiciel peut aussi être installé en tant que service géré par le système d’exploitation, par exemple pour qu’il puisse être lancé en même temps que lui.

Il est possible de configurer de cette manière le fichier JAR créé sous Spring Boot sur toutes les plateformes prises en charge par le logiciel WIPO Sequence, à savoir Windows, Linux et Mac OS.

Le guide ci-dessous présente une méthode détaillée permettant de créer un service système qui exécute le fichier JAR sous chaque système d’exploitation. Il contient en outre des informations sur la manière de configurer les différentes options du service et d’exécuter l’application :

https://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html

[2] https://docs.oracle.com/javase/8/docs/technotes/tools/windows/java.html#BABHDABI

2.2. Déploiement en tant que service Web WAR

En ce qui concerne le second type de fichier binaire proposé, le paquetage WAR peut être déployé sur un serveur d'application Java existant, comme par exemple Apache Tomcat 8.5.

Note : il faut disposer d'un conteneur compatible avec Servlet 3.1.

Les instructions ci-dessous concernent le déploiement sur un serveur d'applications Tomcat. Ici, "$TOMCAT_ROOT" désigne le dossier racine du serveur Tomcat; cette valeur doit être remplacée par le chemin réel :

a) Arrêter le serveur : "$TOMCAT_ROOT\bin\catalina.bat stop"

b) Copier le fichier WAR dans "$TOMCAT_ROOT\webapps\wipo-sequence-validator.war"

c) Lancer le serveur : "$TOMCAT_ROOT\bin\catalina.bat start"

Note : comme Java ne garantit pas l'emploi du codage UTF-8, il faut indiquer "UTF-8" dans le fichier de propriétés système "file.encoding" qui est utilisé au démarrage du serveur d'applications. Ce paramétrage peut être effectué de la manière suivante : -D"file.encoding=UTF-8"

On peut accéder à l'API du logiciel au moyen de l'application Swagger UI, comme indiqué plus haut :

http://host-name:8080/wipo-sequence-validator/swagger-ui.html

On peut accéder à l'API du logiciel depuis les points d'extrémité suivants :

http://[host-name:::]:8080/wipo-sequence-validator/api/[version]/status

http://[host-name:::]:8080/wipo-sequence-validator/api/[version]/validate

L'office de propriété intellectuelle doit apporter les modifications suivantes à cette adresse :

- [host-name] doit être remplacé par le nom du serveur hôte; et
- [version] doit être remplacé par le numéro de version de l'API (p. ex. v1.0).

Par défaut, le logiciel se lance sur le port 8080 du serveur. Pour changer le numéro de port, il faut modifier le fichier de configuration de la Tomcat en suivant les instructions ci-dessous :

https://tomcat.apache.org/tomcat-8.5-doc/config/http.html#Common_Attributes

Par défaut, le logiciel utilise les paramètres mémoire de la machine virtuelle de Java (JVM) définis par défaut. La taille du tas (heap size) maximale par défaut est un quart de la mémoire physique disponible.

Pour modifier la taille du tas maximale, il faut utiliser l'option "-Xmx" lorsque l'exécution est lancée en ligne de commande, comme indiqué plus haut dans la section 2.1.

2.2.1. Rapport de vérification

Le rapport de vérification généré par le logiciel est au format XML et suit le modèle ci-dessous :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<verificationReport productionDate="YYYY-MM-DD" sourceFileName="[ST.26 filename]">
   <verificationMessages>
      <message>
         <severity>[ERROR | WARN | XML_WARN |  XML_ERROR]</severity>
         <dataElement>[ST.26 element]</dataElement>
         <detectedSequence>[Sequence ID]</detectedSequence>
         <detectedValue>[value]</detectedValue>
         <messageKey>[Message key]</messageKey>
         <params>
                <param key="param key">Param value</param>
         <params>
         <localizedMessage> [Localized message] localizedMessage>
      </message>

...

   </verificationMessages>
</verificationReport>
```

Un exemple de ce rapport de vérification est proposé dans l'annexe I du présent manuel avec les valeurs autorisées pour les balises dans l'annexe 3.

2.2.2. Demande du point d'extrémité de rappel

La demande présentée par le point d'extrémité de rappel au logiciel doit contenir les paramètres suivants, qui définissent les emplacements des fichiers et le processus de validation :

```
{
  "currentApplicationNumber": "string",
  "currentSEQLVersionNumber": "string",
  "parentApplicationNumber": "string",
  "parentSEQLVersionNumber": "string",
  "seqlInputLocation": "C:/temp/valid2Warning.xml",
  "verificationReportOutputPath": "C:/temp/report.xml",
  "nameFile": "valid2Warning.xml",
  "type": "full"
}
```

Le champ "seqlInputLocation" de la demande de validation doit être paramétré de manière à indiquer le chemin du listage des séquences XML à valider. Si l'office laisse ce champ vide, le logiciel va tenter de valider le fichier XML portant le nom "nameFile" et situé dans le dossier par

défaut “Inbox”. Le paramètre “nameFile” indique le nom du fichier de listage des séquences à valider.

Le champ “verificationReportOutputPath” de la demande indique l’emplacement du fichier contenant le rapport de vérification (.xml) généré par le logiciel. Si l’utilisateur laisse ce champ vide ou s’il indique un chemin non valable, le rapport de vérification est enregistré dans le dossier par défaut “Reports”.

2.2.2.1. Format de la demande du point d’extrémité de rappel

Si la propriété “api.URL” est configurée, le logiciel va tenter d’envoyer les résultats de la validation au point d’extrémité se trouvant à l’URL indiquée.

Pour communiquer avec le logiciel, le point d’extrémité de rappel doit respecter la convention de service Web suivante (YAML) :

```yaml
openapi: 3.0.0
info:
description: Callback for the WIPO Sequence Validator
version:
title: WIPO Sequence Validator Callback
paths:
/api/validator/callback:
 post:
  summary: Return the generated contract
  operationId: callbackUsingPOST
  requestBody:
   content:
    application/json:
     schema:
      $ref: "#/components/schemas/ServiceRequest”
   description: request
   required: true
  responses:
   "200":
    description: OK
   "201":
    description: Created
   "401":
    description: UNAUTHORIZED
   "403":
    description: FORBIDDEN
   "404":
    description: ELEMENT NOT FOUND
   "500":
    description: INTERNAL ERROR SERVER
  deprecated: false
servers:
- url: //localhost:8080/
```

```
components:
schemas:
 Error:
  type: object
  required:       – code
    – message
  properties:
    code:
      type: string
      example: INVALID_VALIDATION_TYPE
      description: error code
    message:
      type: string
      description: error message
    moreInfo:
      type: string
      description: extended info on the error
  title: Error
 MapEntry:
  type: object
  properties:
    key:
      type: string
      xml:
        name: key
        attribute: true
        wrapped: false
    value:
      type: string
  title: MapEntry
  xml:
    name: Parameter
    attribute: false
    wrapped: false
 ServiceRequest:
  type: object
  properties:
    currentApplicationNumber:
      type: string
    currentSEQLVersionNumber:
      type: string
    elapsedTime:
      type: integer
      format: int64
    endTime:
      type: string
```

```
    errorSummary:
      type: array
      items:
        $ref: "#/components/schemas/VerificationMessage”
    httpStatus:
      type: string
    parentApplicationNumber:
      type: string
    parentSEQLVersionNumber:
      type: string
    processID:
      type: string
    seqIDQuantity:
      type: integer
      format: int32
    seqInputQuantity:
      type: integer
      format: int32
    seqlType:
      type: string
    startTime:
      type: string
    totalErrorQuantity:
      type: integer
      format: int32
    totalWarningQuantity:
      type: integer
      format: int32
    verificationReportOutputPath:
      type: string
  title: ServiceRequest
VerificationMessage:
  type: object
  properties:
    dataElement:
      type: string
      xml:
        name: DataElement
        attribute: false
        wrapped: false
    detectedSequence:
      type: string
      xml:
        name: DetectedSequence
        attribute: false
        wrapped: false
```

```
    index:
      type: integer
      format: int32
    key:
      type: string
      xml:
        name: MessageKey
        attribute: false
        wrapped: false
    locmessage:
      type: string
      xml:
        name: LocalizedMessage
        attribute: false
        wrapped: false
    params:
      type: object
      additionalProperties:
        type: string
    paramsForXML:
      type: array
      xml:
        name: ParameterBag
        attribute: false
        wrapped: true
      items:
        $ref: "#/components/schemas/MapEntry"
    reportValue:
      type: string
      xml:
        name: DetectedValue
        attribute: false
        wrapped: false
    sequenceIDNumber:
      type: string
    type:
      type: string
      xml:
        name: Severity
        attribute: false
        wrapped: false
  title: VerificationMessage
  xml:
    name: VerificationMessage
    attribute: false
    wrapped: false
```

En outre, la demande doit être un objet JSON ayant la structure suivante :

```
{
  "currentApplicationNumber": "string",
  "currentSEQLVersionNumber": "string",
  "elapsedTime": 0,
  "endTime": "string",
  "errorSummary": [
    {
      "dataElement": "string",
      "detectedSequence": "string",
      "index": 0,
      "key": "string",
      "locmessage": "string",
      "params": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "paramsForXML": [
        {
          "key": "string",
          "value": "string"
        }
      ],
      "reportValue": "string",
      "sequenceIDNumber": "string",
      "type": "string"
    }
  ],
  "httpStatus": "string",
  "parentApplicationNumber": "string",
  "parentSEQLVersionNumber": "string",
  "processID": "string",
  "seqIDQuantity": 0,
  "seqInputQuantity": 0,
  "seqlType": "string",
  "startTime": "string",
  "totalErrorQuantity": 0,
  "totalWarningQuantity": 0,
  "verificationReportOutputPath": "string"
}
```

Voici un exemple d'instance JSON à envoyer au point d'extrémité extérieur ayant appelé le logiciel :


```
{
   "processID": "1608194222169dvVE",
   "seqlType": "ST.26",
   "httpStatus": "SUCCESS",
   "currentApplicationNumber": "string",
   "currentSEQLVersionNumber": "string",
   "parentApplicationNumber": "string",
   "parentSEQLVersionNumber": "string",
   "verificationReportOutputPath": "C:/temp/report.xml",
   "startTime": "2020-12-17 09:36:54.000000",
   "endTime": "2020-12-17 09:37:26.000607",
   "elapsedTime": 32607,
   "totalWarningQuantity": 1,
   "totalErrorQuantity": 2,
   "seqInputQuantity": 3,
   "seqIDQuantity": 3,
   "errorSummary": [
      {
         "index": 0,
         "reportValue": "",
         "type": "WARNING",
         "params":com.wipo.st26.ipotool.models.ServiceRequest@5887858,
         "key": "X_EARLIEST_PRIO_APPLICATION_ID_MISSING",
         "locmessage": "Earliest priority application information is absent. It must be provided when a pri
ority claim is made to an earlier application.",
         "detectedSequence": "",
         "dataElement": "PROPERTY_NAMES.EARLIEST_PRIORITY_APPLICATION"
      },
      {
         "index": 0,
         "reportValue": "-",
```

```
        "type": "ERROR",
        "params": {},
        "key": "INVENTION_TITLE_MISSING",
        "locmessage": "The invention title is missing. At least one invention title must be entered.",
        "detectedSequence": "",
        "dataElement": "PROPERTY_NAMES.INVENTION_TITLE_BAG"
      },
      {
        "index": 1,
        "reportValue": "-",
        "type": "ERROR",
        "params": {},
        "key": "INVENTION_TITLE_MISSING",
        "locmessage": "The invention title is missing. At least one invention title must be entered.",
        "detectedSequence": "",
        "dataElement": "PROPERTY_NAMES.INVENTION_TITLE_BAG"
    }
 ]
}
```


2.2.2.2. Rapport de vérification

Comme il est indiqué dans la section 2.2.1, après la validation, le rapport de vérification généré est déplacé vers l'emplacement défini dans "verificationReportOutputPath", qui dans notre exemple se trouve ici : "C:/temp/report.xml".

Le contenu de ce rapport est envoyé vers le point d'extrémité de rappel dans le champ "errorSummary" de la demande "ServiceRequest". On trouvera une illustration de ce champ dans les exemples de demande présentés plus haut dans la section 2.2.2.

2.3. Configuration

2.3.1. Paramétrage par défaut

Le logiciel est configuré au moyen d'un fichier de propriétés. Le fichier "application.properties" utilisé par défaut contient les valeurs suivantes :

```
############ WIPO Sequence Validator properties
```

```
## -- FOLDERS --

#Base path to be used by the rest of folders
app.basePath=/temp/ST26/
#Folder to put the files to be processed
app.inboxPath=${app.basePath}inbox/

#Folder to store the ST26 files once validated
app.outboxPath=${app.basePath}outbox/
#Folder to store the validation reports
app.reportsPath=${app.basePath}reports/

#Parent folder for full and formality folders
app.processPath=${app.basePath}process/

#Files in process for a full validation are stored in this folder
app.process.fullPath=${app.processPath}full/
#Files in process for a formality validation are stored in this folder
app.process.formalityPath=${app.processPath}formality/

#Folder to store the parameters
app.paramsPath=${app.basePath}params/

alternativeResourceBasePath=${app.basePath}/alt_resources

#locale used for the localized messages from the verification report
validator_locale=en

#URL of the callback endpoint that will be used for informing of the results of
#    the validation.  If not set or empty, the callback with the results of the
#    validation will not occur
api.URL=http://callbackservice/api/endpoint

## -- Watcher

# These properties control the process looking for files in the folders to be processed
# (see: https://docs.spring.io/spring-framework/docs/current/javadoc-
api/org/springframework/scheduling/concurrent/ThreadPoolTaskExecutor.html)

processing.delay=10000
processing.corePoolSize=5
#Max number of files being validated concurrently
processing.maxPoolSize=10
processing.queueCapacity=1000
```

```
#### Logging (see https://logback.qos.ch/manual/configuration.html)

logging.level.root=info
logging.level.com.wipo=info
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} – %msg%n
```

Pour modifier la valeur des paramètres de ce fichier, il faut utiliser un autre fichier "application.properties". Plusieurs options sont possibles, comme l'indique la documentation de Spring Boot : https://docs.spring.io/spring-boot/docs/2.0.6.RELEASE/reference/html/boot-features-external-config.html#boot-features-external-config-application-property-files

La solution la plus simple consiste à fournir un nouveau fichier "application.properties" qui sera recherché dans les emplacements suivants, dans l'ordre de fonctionnement du logiciel :

a) Un dossier "/config" dans le répertoire en cours *[Note : si le logiciel est déployé par un fichier WAR dans une Tomcat, ce dossier se trouve dans le répertoire "lib", par exemple dans le chemin "/opt/apache-tomcat/lib/config"]*;

b) Le répertoire en cours *[Note : si le logiciel est déployé par un fichier WAR dans une Tomcat, ce dossier se trouve dans le répertoire "lib", par exemple dans le chemin "/opt/apache-tomcat/lib/"];*

c) Un chemin de classe ($classpath) ou un paquetage de configuration; et dans ce cas,

d) Dans la racine $classpath.

Le chemin et le nom du fichier de configuration peuvent aussi être indiqués en spécifiant le paramètre en ligne de commande lorsqu'on lance le logiciel :

a) Pour un déploiement par un fichier JAR :

```
java -D"file.encoding=UTF-8" -jar wipo-sequence-validator.jar – -
spring.config.location=<PATH_TO_FILE>
```

b) Pour un déploiement par un fichier WAR dans une Tomcat, il faut ajouter l'entrée suivante dans les options CATALINA_OPTS :

```
"export CATALINA_OPTS="-Dspring.config.location=<PATH_TO_FILE>"
```

Si l'on déploie le logiciel par un fichier WAR, le nouveau dossier "application.properties" peut aussi être copié dans le dossier "WEB-INF/classes" de l'application Web.

Note : le logiciel doit être redémarré pour que les propriétés définies dans le nouveau fichier "application.properties" soient prises en compte.

2.3.2. Messages en langue locale

Le logiciel peut générer des messages, par exemple dans le rapport de vérification, dans chacune des 10 langues officielles du PCT (allemand, anglais, arabe, chinois, coréen, espagnol, français, japonais, portugais et russe). Par défaut, ces messages sont générés en anglais. Pour configurer le logiciel afin qu'il génère ces messages dans une autre langue, le paramètre "validator_locale" du fichier "application.properties" doit être modifié en indiquant le code de langue souhaité. Par exemple, "validator_locale=es".

2.3.3. Noms d'organismes personnalisés

Pour que les offices puissent employer leurs propres noms d'organismes personnalisés, c'est-à-dire des noms qui ne font pas partie de la liste originale prédéfinie de noms d'organismes, ils peuvent fournir une liste d'organismes personnalisés en créant un nouveau fichier appelé "custom_organism.json" et en le plaçant dans le dossier "alternativeResourceBasePath". Ce fichier doit avoir la structure suivante :

```
[
   {"value":"Custom Organism Sample"},
   {"value":"Custom Organism Sample 2"}
]
```

Note : à la différence de la liste prédéfinie de noms d'organismes, tous les organismes sont contenus dans un seul fichier JSON; ils ne sons pas répartis dans un fichier JSON pour chaque lettre de l'alphabet.

2.3.4. Fichiers DTD du format ST.26 servant de référence

Par défaut, le logiciel utilise la version la plus récente de la DTD du format ST.26 de l'OMPI. La version actuelle du logiciel exploite la version 1.3 de cette DTD[3].

Cette version de la DTD du format ST.26, qui est la plus récente, est intégrée dans la bibliothèque du logiciel qui se trouve dans le dossier "/src/main/resources" du code source (ce chemin est celui qui est défini dans le fichier JAR ou WAR). Elle est indiquée en référence dans le fichier "catalog.xml" qui se trouve dans le même dossier, comme illustré ci-dessous :

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
  <public publicId="-//WIPO//DTD Sequence Listing 1.3//EN" uri="ST26SequenceListing_V1_3.dtd"/>
</catalog>
```

On trouvera ci-après les instructions permettant d'intégrer une nouvelle DTD. La version de la DTD employée au cours de la validation est celle qui est définie dans la déclaration DOCTYPE du fichier XML. Dans un premier temps, le système utilise le champ "publicId" pour déterminer l'emplacement de la DTD à utiliser. Si le catalogue ne contient pas de champ "publicId", le système tente de trouver la DTD dans le dossier racine dans lequel le processus Java est en cours d'exécution.

Comment indiquer une autre version de DTD pour la validation

Pour pouvoir valider des fichiers ST.26 par rapport à une version plus ancienne de la DTD de ce format, le fichier DTD concerné doit être accessible au logiciel.

À cette fin, deux méthodes sont possibles :

a) Décompresser le fichier JAR et ajouter une référence à la DTD supplémentaire ou alternative du format ST.26 dans le dossier "src/main/resources". Modifier le fichier "catalog.xml" et ajouter une nouvelle entrée définissant la DTD supplémentaire, ou modifier l'entrée existante.

[3] Valable au 14 janvier 2021.

Par exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
<public publicId="-//WIPO//DTD Sequence Listing 1.2//EN" uri="ST26SequenceListing_V1_2.dtd"/>
<public publicId="-//WIPO//DTD Sequence Listing 1.3//EN" uri="ST26SequenceListing_V1_3.dtd"/>
</catalog>
```

b) Au lieu de modifier le fichier JAR, on peut suivre les étapes suivantes :

i) Copier le fichier “catalog.xml” et toutes les DTD dans un dossier local;
ii) Modifier le fichier “catalog.xml” pour ajouter une référence à la DTD ST.26 supplémentaire; et
iii) Définir cette propriété système Java au lancement : “xml.catalog.files=<path_to_catalog.xml>"

Note : si la Tomcat tourne sous Windows, on peut également le faire en ajoutant cette variable d’environnement :

```
set "JAVA_OPTS=%JAVA_OPTS%
 -Dxml.catalog.files=C:\\temp\\tomcat\\sharedclasspath\\catalog.xml"
```

[IMPORTANT : le fait d’ajouter une version différente de la DTD ST.26 permet d’effectuer une validation “de formalité” du fichier XML par rapport à cette nouvelle DTD, mais pour effectuer une validation “complète”, il faudra probablement modifier le code source pour définir les règles de vérification. Il est donc recommandé de n’utiliser plusieurs DTD que pour effectuer une validation “de formalité”.]

3. API REST du logiciel

La présente section contient les spécifications des cas d'utilisation de l'API du logiciel. Il existe trois services ou cas d'utilisation :

a) Valider un fichier dans le dossier "Inbox";

b) Valider un fichier intégré à une demande; et

c) Demander le statut d'une validation.

Les spécifications de l'API permettant d'exécuter ces services (API complète conforme à la spécification OAS 3.0 [fichier YAML]) sont indiquées en détail dans l'annexe II.

3.1. Valider un fichier au format ST.26 de l'OMPI

Mappage de la demande	/api/v1/validate
Méthode	POST
Prend en entrée	application/json
Produit en sortie	application/json
Fonctionnement	Demande la validation d'un fichier ST.26 existant dans le dossier "Inbox". Renvoie une "verificationID" unique permettant d'obtenir le statut de la demande de validation.

Demande	{
	"currentApplicationNumber": "string",
	"currentSEQLVersionNumber": "string",
	"parentApplicationNumber": "string",
	"parentSEQLVersionNumber": "string",
	"seqlInputLocation": "string" (emplacement du fichier Input.xml),
	"verificationReportOutputPath": "string" (destination du fichier report.xml),
	"nameFile": "file.xml",
	"type": "string" (valeurs possibles : full | formality),
	}

Réponses	• '202' : "Accepted" (Accepté). Le fichier ST.26 a obtenu la validation de formalité et sa vérification par rapport aux règles officielles a commencé. Ce code contiendra aussi un message de réponse indiquant un code unique qui permet d'obtenir le rapport de vérification ("verificationID"). Le fichier ST.26 est déplacé vers le dossier "Process" pour être traité.

	• '400' : “Bad request” (Demande erronée). La demande REST n’est pas bien formée ou le fichier ST.26 n’a pas obtenu la validation XML. Ce code sera complété par un message de réponse contenant des informations détaillées sur l’erreur. • '404' : “Not Found” (Introuvable). Le fichier ST.26 n’a pas été trouvé dans le dossier “Inbox”. • '500' : “Internal Server Error” (Erreur de serveur interne). Une erreur interne s’est produite. Ce code sera complété par un message de réponse contenant des informations détaillées sur l’erreur.
Condition préalable	Le fichier ST.26 doit avoir été copié dans le dossier “Inbox” défini par l’utilisateur.
Condition postérieure	Le fichier ST.26 est déplacé vers l’emplacement indiqué dans verificationReportOutputPath ou vers le dossier “Outbox” situé à l’emplacement suivant : “/[outbox]/[verificationID]/[file.xml]” Le rapport de vérification est généré à l’emplacement suivant : “/[reports]/ [verificationID]/report_[file.xml]”

Définition correspondante selon la spécification OAS 3.0 [en spécification YAML]

```
 /api/v1.0/validate:
post:
  tags:
    - validation-controller
  summary: 'Request the validation of an existing ST26 file in the inbox folder. Re
turns a unique verificationID for retrieving the status of the validation request'
  operationId: validationFileUsingPOST
  consumes:
    - application/json
  produces:
    - application/json
  parameters:
    -
      in: body
      name: request
      description: 'ST26 File name for validation'
      required: false
      schema:
        $ref: '#/definitions/ValidationRequest'
  responses:
    '200':
      description: OK
```

```
      schema:
        $ref: '#/definitions/ValidationResponse'
    '201':
      description: Created
    '202':
      description: 'Accepted. The ST26 file passed the formal validation and their
verification has started. This code will be complemented with a response message in
dicating a unique code for retrieving the verification report (verificationID). ST2
6 file is moved to the process folder for processing'
      schema:
        $ref: '#/definitions/ValidationResponse'
    '400':
      description: 'Bad request. The REST request was not well formed or the ST26 f
ile did not pass the XML validation. This code will be complemented with a response
 message indicating the detail of the error'
    '401':
      description: Unauthorized
    '403':
      description: Forbidden
    '404':
      description: 'File not found. The ST26 file was not found in the Inbox Folder
'
    '500':
      description: 'Server Error. An internal error happened. This code will be com
plemented with a response message indicating the detail of the error'
  deprecated: false
```

3.2. Statut de validation de la demande

Mappage de la demande	/api/v1/status
Méthode	POST
Prend en entrée	application/json
Produit en sortie	application/json
Fonctionnement	Demande le statut de la validation d'un fichier ST.26 particulier

Demande	{
	verificationID : {type: string}
	}

Réponses	a) '200' : Success (Succès). Ce code contient aussi un message de réponse indiquant un identifiant unique ainsi que le statut du processus de vérification, qui peut prendre l'une des valeurs suivantes : ○ "RUNNING" (En cours) : le fichier est en cours de traitement ○ "FINISHED-VALID" (Achevé – Valable) : le fichier a obtenu la validation de formalité et le résultat de cette validation peut être consulté dans le dossier des rapports ○ "FINISHED-INVALID" (Achevé – Non valable) : le processus est achevé mais le fichier n'a pas obtenu la validation de formalité. Le résultat du processus de validation peut être consulté dans le dossier des rapports ○ "NOT_FOUND" : l'identifiant "verificationID" n'a pas été trouvé ○ "VERIFICATION_ID_ERROR" : l'identifiant "verificationID" n'a pas été intégré dans la demande b) '400' : "Bad request" (Demande erronée). La demande REST n'est pas bien formée c) '500' : "Internal Server Error" (Erreur de serveur interne). Une erreur interne s'est produite. Ce code sera accompagné d'un message de réponse contenant des informations détaillées sur l'erreur.
Condition postérieure	Le logiciel doit indiquer le statut de la validation.

Hypothèses	-

Définition correspondante selon la spécification OAS 3.0 [en spécification YAML]

```
paths:
  /api/v1.0/status:
    post:
      tags:
        - validation-controller
      summary: 'Request the validation status for a specific ST26 File'
      operationId: getStatusUsingPOST
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        -
          in: body
          name: request
          description: 'ST26 File name Object for validation status'
          required: false
          schema:
            $ref: '#/definitions/ValidationStatusRequest'
      responses:
        '200':
          description: 'This code will be complemented with a response message indi
cating the Status of the verification process: RUNNING (the file is being processed
) FINISHED-
VALID (the file passed the formality validation and the result of the validation is
 available in the reports folder) FINISHED-
INVALID (the file passed the formality validation and the result of the validation
is available in the reports folder)'
          schema:
            $ref: '#/definitions/ValidationStatusResponse'
        '201':
          description: Created
        '400':
          description: 'Bad request. The REST request was not well formed'
        '401':
          description: Unauthorized
        '403':
          description: Forbidden
        '404':
          description: 'Not Found'
        '500':
```

```
        description: 'Server Error. An internal error happened. This code will be
complemented with a response message indicating the detail of the error'
    deprecated: false
```

[L’annexe I suit]

Annexe I : Exemple de rapport de vérification

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerificationReport productionDate="2020-12-17"
sourceFileName="valid2Warning.xml">
 <VerificationMessageBag>
     <VerificationMessage>
         <Severity>WARNING</Severity>
         <DataElement>PROPERTY_NAMES.EARLIEST_PRIORITY_APPLICATION</DataElement>
         <DetectedSequence></DetectedSequence>
         <DetectedValue></DetectedValue>
         <MessageKey>X_EARLIEST_PRIO_APPLICATION_ID_MISSING</MessageKey>
         <ParameterBag/>
         <LocalizedMessage>Earliest priority application information is absent.
It must be provided when a priority claim is made to an earlier
application.</LocalizedMessage>
     </VerificationMessage>
     <VerificationMessage>
         <Severity>ERROR</Severity>
         <DataElement>PROPERTY_NAMES.INVENTION_TITLE_BAG</DataElement>
         <DetectedSequence></DetectedSequence>
         <DetectedValue>-</DetectedValue>
         <MessageKey>INVENTION_TITLE_MISSING</MessageKey>
         <ParameterBag/>
         <LocalizedMessage>The invention title is missing.  At least one
invention title must be entered.</LocalizedMessage>
     </VerificationMessage>
     <VerificationMessage>
         <Severity>ERROR</Severity>
         <DataElement>PROPERTY_NAMES.INVENTION_TITLE_BAG</DataElement>
         <DetectedSequence></DetectedSequence>
         <DetectedValue>-</DetectedValue>
         <MessageKey>INVENTION_TITLE_MISSING</MessageKey>
         <ParameterBag/>
         <LocalizedMessage>The invention title is missing.  At least one
invention title must be entered.</LocalizedMessage>
     </VerificationMessage>
 </VerificationMessageBag>
</VerificationReport>
```

[L’annexe II suit]

Annexe II : Spécification complète de l'API (YAML)

```
openapi: 3.0.0
info:
  description: API for the WIPO Sequence Validator
  version: "1.0"
  title: WIPO Sequence Validator API
tags:
  - name: validation-controller
    description: Validation Controller
paths:
  /api/v1.0/status:
    post:
      tags:
        - validation-controller
      summary: Request the validation status for a specific ST26 File
      operationId: getStatusUsingPOST
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/ValidationStatusRequest"
        description: ST26 File name Object for validation status
      responses:
        "200":
          description: "This code will be complemented with a response message
indicating
            the Status of the verification process: RUNNING (the file is being
            processed) FINISHED-VALID (the file passed the formality validation
            and the result of the validation is available in the reports folder)
            FINISHED-INVALID (the file passed the formality validation and the
            result of the validation is available in the reports folder)"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/ValidationStatusResponse"
        "201":
          description: Created
        "400":
          description: Bad request. The REST request was not well formed
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: Not Found
        "500":
          description: Server Error. An internal error happened. This code will
be
            complemented with a response message indicating the detail of the
            error
      deprecated: false
  /api/v1.0/validate:
    post:
      tags:
        - validation-controller
```

```
      summary: Request the validation of an existing ST26 file in the inbox
folder.
        Returns a unique verificationID for retrieving the status of the
        validation request
      operationId: validationFileUsingPOST
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/ValidationRequest"
        description: ST26 File name for validation
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/ValidationResponse"
        "201":
          description: Created
        "202":
          description: Accepted. The ST26 file passed the formal validation and
their
            verification has started. This code will be complemented with a
            response message indicating a unique code for retrieving the
            verification report (verificationID). ST26 file is moved to the
            process folder for processing
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/ValidationResponse"
        "400":
          description: Bad request. The REST request was not well formed or the
ST26 file
            did not pass the XML validation. This code will be complemented with
            a response message indicating the detail of the error
        "401":
          description: Unauthorized
        "403":
          description: Forbidden
        "404":
          description: File not found. The ST26 file was not found in the Inbox
Folder
        "500":
          description: Server Error. An internal error happened. This code will
be
            complemented with a response message indicating the detail of the
            error
      deprecated: false
servers:
  - url: //localhost:8080/
components:
  schemas:
    ValidationRequest:
      type: object
    required:
      - nameFile
      - type
```

```
      - verificationReportOutputPath
    properties:
      nameFile:
        type: string
        example: file.xml
        description: 'File Name Validation'
      type:
        type: string
        example: 'full or formality'
        description: 'Type of validation'
      currentApplicationNumber:
        type: string
        example: 1.3
        description: 'The application number associated with the sequence listin
g'
      currentSEQLVersionNumber:
        type: string
        example: 1.2
        description: 'the version number of this sequence listing (internally as
signed by an Office)'
      parentApplicationNumber":
        type: string
        example: WIPO-1234
        description: 'Any associated parent application'
      parentSEQLVersionNumber:
        type: string
        example: 1.1
        description: 'The version number of the parent’s sequence listing'

      seqlInputLocation:
        type: string
        example: /st26/inbox/file.xml
        description: 'Contains the path of the input xml file to be validated'
      verificationReportOutputPath:
        type: string
        example: /st26/outbox/file.xml
        description: 'Will contain the destination path of the report.xml genera
ted by the tool.'

      title: ValidationRequest
      description: Class representing a response Validation status File by the
application.
    ValidationResponse:
      type: object
      required:
        - verificationID
      properties:
        errorMsg:
          type: string
        verificationID:
          type: string
          example: 1552208288697FNc2
          description: verificationID
      title: ValidationResponse
      description: Class representing a response Validation ST26 File by the
application.
    ValidationStatusRequest:
      type: object
```

```
      required:
        - verificationID
      properties:
        verificationID:
          type: string
          example: 1552208288697FNc2
          description: verificationID
      title: ValidationStatusRequest
      description: Request of the validation status of an ST26 File
    ValidationStatusResponse:
      type: object
      required:
        - status
      properties:
        status:
          type: string
          example: RUNNING - FINISHED_VALID - FINISHED_INVALID
          description: Valdation Status File
        reportPath:
          type: string
          example: /st26/reports/1552208288697FNc2/report_file.xml
          description: ReportFilePath
      title: ValidationStatusResponse
      description: Response with the validation status for a specific
verificationID.
```


[L’annexe III suit]

Annexe III : Noms des propriétés (JSON)


```
{
    "featureKey": "PROPERTY_NAMES.FEATURE_KEY",
    "featureLocation": "PROPERTY_NAMES.FEATURE_LOCATION",
    "featureQuals": "PROPERTY_NAMES.FEATURE_QUALS",
    "qualifierName": "PROPERTY_NAMES.QUALIFIER_NAME",
    "qualifierValue": "PROPERTY_NAMES.QUALIFIER_VALUE",
    "qualifierTranslatedValue": "PROPERTY_NAMES.QUALIFIER_TRANSLATED_VALUE",
    "qualifierId": "PROPERTY_NAMES.QUALIFIER_ID",
    "length": "PROPERTY_NAMES.SEQUENCE_LENGTH",
    "INSDSeqMoltype": "PROPERTY_NAMES.SEQ_MOL_TYPE",
    "INSDQualifierMolType": "PROPERTY_NAMES.QUAL_MOL_TYPE",
    "organism": "PROPERTY_NAMES.ORGANISM",
    "featureTable": "PROPERTY_NAMES.FEATURE_TABLE",
    "INSDSeqSequence": "PROPERTY_NAMES.SEQ_SEQUENCE",
    "division": "PROPERTY_NAMES.DIVISION",
    "sequenceIDNumber": "PROPERTY_NAMES.SEQUENCE_ID_NUMBER",
    "applicationIdentification": "PROPERTY_NAMES.APPLICANT_IDENTIFICATION",
    "applicationIdentification.filingDate": "PROPERTY_NAMES.APPLICANT_IDENTIFICATIO
N",
    "applicantFileReference": "PROPERTY_NAMES.APPLICANT_FILE_REFERENCE",
    "earliestPriorityApplicationIdentification": "PROPERTY_NAMES.EARLIEST_PRIORITY_
APPLICATION",
    "earliestPriorityApplicationIdentification.filingDate": "PROPERTY_NAMES.EARLIES
T_PRIORITY_APPLICATION",
    "applicantName": "PROPERTY_NAMES.APPLICANT",
    "applicantName.name": "PROPERTY_NAMES.APPLICANT_NAME",
    "applicantName.languageCode": "PROPERTY_NAMES.APPLICANT_LANGUAGE_CODE",
    "applicantName.nameLatin": "PROPERTY_NAMES.APPLICANT_NAME_LATIN",
    "inventorName": "PROPERTY_NAMES.INVENTOR",
    "inventorName.name": "PROPERTY_NAMES.INVENTOR_NAME",
    "inventorName.languageCode": "PROPERTY_NAMES.INVENTOR_LANGUAGE_CODE",
    "inventorName.nameLatin": "PROPERTY_NAMES.INVENTOR_NAME_LATIN",
    "inventionTitleBag": "PROPERTY_NAMES.INVENTION_TITLE_BAG",
    "priorityInformationBag": "PROPERTY_NAMES.PRIORITY_INFORMATION_BAG",
    "sequenceDataBag": "PROPERTY_NAMES.SEQUENCE_DATA_BAG",
    "sequenceTotalQuantity": "PROPERTY_NAMES.SEQUENCE_TOTAL_QUANTITY",
    "fileName": "PROPERTY_NAMES.FILE_NAME",
    "dtdVersion": "PROPERTY_NAMES.DTD_VERSION",
    "softwareName": "PROPERTY_NAMES.SW_NAME",
    "softwareVersion": "PROPERTY_NAMES.SW_VERSION",
    "productionDate": "PROPERTY_NAMES.PRODUCTION_DATE",
    "originalFreeTextLanguageCode": "PROPERTY_NAMES.ORIGINAL_FREE_TEXT_LANGUAGE_COD
E",
```

```
    "nonEnglishFreeTextLanguageCode": "PROPERTY_NAMES.NON_ENGLISH_FREE_TEXT_LANGUAG
E_CODE",

}
```

[Fin du document]