OMPI
ORGANIZACIÓN MUNDIAL
DE LA PROPIEDAD
INTELECTUAL

WIPO Sequence Validator
Versión 2.1.0

Manual de uso

El propósito de este documento es ayudar a las Oficinas de Propiedad Intelectual a implementar el servicio web de WIPO Sequence Validator, así como proporcionar apoyo para configurar la herramienta.

Índice

1. Introducción

La principal utilidad de WIPO Sequence Validator (en adelante, 'Validador' o 'la herramienta') es proporcionar a las Oficinas de Propiedad Intelectual (Oficinas de PI) un servicio web para validar archivos XML a fin de garantizar su conformidad con la Norma ST.26 de la OMPI. Para generar una lista de secuencias se podrá utilizar la herramienta que se considere más adecuada, aunque si se genera con la aplicación de escritorio WIPO Sequence será compatible con la Norma ST.26 de la OMPI.

La finalidad de este documento es explicar la estructura, la implementación, la configuración y el sistema de archivos de la herramienta. Para consultar preguntas sobre resolución de problemas, los usuarios se pueden dirigir a la wiki, en la siguiente dirección: https://www3.wipo.int/confluence/display/ST26software/Validator+Troubleshooting.

1.1. Descripción general del flujo de trabajo del Validador

La herramienta permite realizar las cuatro operaciones siguientes:

- Validar un archivo para verificar su conformidad con la Norma ST.26 de la OMPI;
- Solicitar información sobre el estado de una validación en curso;
- Actualizar los archivos de configuración (solo la administración de la Oficina de PI); y
- Llamar a un punto final de devolución de llamada con el resultado del proceso de validación una vez completado.

Nota: el punto final de devolución de llamada[1] no es establecido por el Validador sino por la Oficina que desarrolle y configure el servicio.

La herramienta se implementa con un archivo JAR, que puede ejecutarse como servicio web, o con un archivo WAR, que puede ejecutarse en un servidor Tomcat.

En ambos casos, para validar una lista de secuencias conforme a la Norma ST.26 de la OMPI, la herramienta utiliza archivos de un sistema de archivos local, genera un informe de verificación con los resultados de la validación y, opcionalmente, devuelve los resultados del proceso de validación —el informe de verificación— llamando a un punto final de devolución de llamada.

A continuación se describen los principales pasos del flujo de trabajo del Validador:

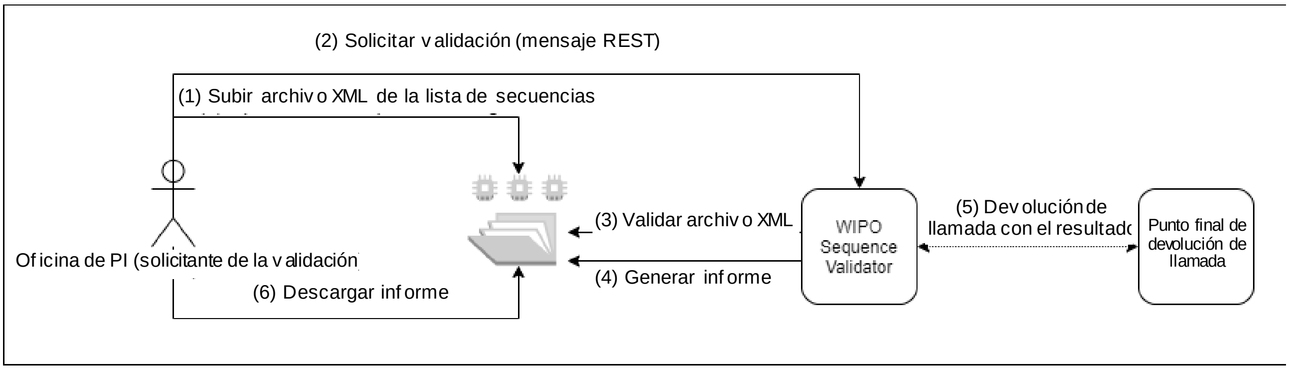
- El sistema de TI de la Oficina de PI en cuestión guarda el archivo XML que se va a validar conforme a la Norma ST.26 de la OMPI en una carpeta predeterminada de entrada (carpeta “Inbox”) o en la especificada en el pedido.
- El sistema de TI de la Oficina de PI envía un pedido HTTP POST solicitando la validación del archivo. Dependiendo de la configuración, podrá solicitar una validación del archivo completa (“full”) o de forma (“formality”). En el proceso de validación de forma se comprobará si el archivo es un archivo XML y se validará tomando como referencia la DTD prevista en la Norma ST.26. En el proceso de validación completa se verificará si el archivo en formato ST.26 cumple las normas operativas derivadas del contenido de la Norma ST.26, además de realizarse la validación de forma.

  Nota: Se recomienda la validación de forma solo para el sistema de aceptación de pedidos en línea, ya que puede realizarse de forma sincronizada, mientras que la validación

[1] Se entenderá por un punto final de devolución de llamada una dirección única identificada por un URI a la que se pueden enviar mensajes de pedidos.

completa se recomienda para la ejecución por lotes, dado que requerirá mucho más tiempo.

- Una vez completada la validación, se proporcionará una respuesta en la que se indicará si el archivo ha superado o no la validación de forma y, en el caso de que el sistema de TI de la Oficina de PI haya seleccionado la validación completa, también si el proceso de validación de las normas operativas se ha iniciado correctamente.
- Cuando el Validador realiza una validación completa, accede al archivo XML en la carpeta “Inbox” e inicia el proceso de validación de las normas operativas. Seguidamente:
- El Validador genera un archivo XML de informe —el informe de verificación— en la carpeta especificada de salida y mueve el archivo XML validado conforme a la Norma ST.26 de la OMPI a la carpeta “Outbox”.
- Una vez completado el proceso de validación de las normas operativas, el Validador llama al punto final de devolución de llamada, si está configurado, y se incluye en el pedido información adicional relacionada con el proceso de validación. En el apartado 4 se describe la estructura del pedido y se incluyen algunos datos de ejemplo.
- El punto final de devolución de llamada no devolverá nada o devolverá un código que indique que el proceso se ha completado correctamente (sin errores). *[Nota: este paso solo se realiza si el servicio web externo está disponible y se ha configurado la llamada en el Validador]*. También es necesario que el Validador y el punto final de devolución de llamada estén conectados. Como ya se ha mencionado, el servicio web externo no forma parte del Validador, sino que serán las Oficinas las que lo desarrollen y configuren de acuerdo con el contrato que se define más adelante.
- El sistema de TI de la Oficina de PI puede acceder al informe de verificación guardado en la carpeta “Reports”.



Nota: WIPO Sequence Validator es compatible con la [Norma ST.90 de la OMPI] para el tratamiento y la comunicación de datos de propiedad intelectual mediante interfaces de programación de aplicaciones (API) para servicios web.

1.2. Definición de la estructura del sistema de archivos del Validador

El sistema de archivos utilizado por el Validador consta de cinco carpetas principales:

- **Carpeta “Inbox”**: Carpeta local en la que las Oficinas de PI guardan los archivos que se van a validar conforme a la Norma ST.26 de la OMPI.

- **Carpeta "Process"**: Carpeta local en la que los archivos de la carpeta "Inbox" se almacenan temporalmente durante el proceso. Contiene dos subcarpetas:
    - **La carpeta "Full validation"**, en la que se guardan los archivos que van a ser sometidos a una validación completa; y
    - **La carpeta "Formality validation"**, en la que se almacenan los archivos que van a ser sometidos a una validación de forma.
- **Carpeta "Outbox"**: Carpeta local en la que la aplicación almacena el archivo de origen, una vez completada la validación conforme a la Norma ST.26 de la OMPI.
- **Carpeta "Reports"**: Carpeta local en la que se guarda el archivo de informe de verificación con los resultados de la validación.
- **Carpeta "Params"**: Carpeta local en la que se guarda un archivo JSON (.json) con todos los parámetros de validación extraídos del pedido de validación que se proporcionarán para una validación asincrónica en profundidad.

A continuación figura un ejemplo de estructura del sistema de archivos:

[IMPORTANTE: Por defecto, el directorio /temp/ST26 se debe ubicar en el directorio padre donde se encuentra la herramienta. Por ejemplo, si el archivo JAR o WAR se ubica en C:/dev, la estructura de carpetas creada debe ser C:/temp/ST26/…]

2. Implementación de WIPO Sequence Validator

Como se ha indicado anteriormente, el Validador se proporciona en uno de los dos formatos de archivo binario que se indican a continuación. Dependiendo del tipo de sistema en el que la Oficina quiera implementar el Validador, será preferible seleccionar un tipo de archivo binario u otro.

Los dos formatos de archivo binario en el que se ofrece el Validador son:

- **Archivo binario JAR de Spring Boot**: archivo JAR ejecutable. Requiere que Java 8 esté instalado.
- **Archivo binario WAR:** archivo que se ejecuta en un contenedor servlet. Se requiere un servidor de aplicaciones compatible con Spring Boot 2 y Servlet Spec 3.1+, como Tomcat 8.5.

En las siguientes secciones se explica la implementación del Validador como una aplicación Spring Boot o como un archivo WAR en un servidor de aplicaciones Java.

2.1. Implementación del Validador como un archivo JAR de Spring Boot

El archivo JAR de Spring Boot con su servidor embebido permite la implementación de la API del Validador sin necesidad de recurrir a un servidor externo. Esto simplifica enormemente la configuración y la implementación a nivel de sistema.

El servidor embebido se ejecuta con el siguiente comando.

Nota: Se requiere que Java 8 esté instalado en el servidor. Para garantizar que Java permita abrir archivos en formato UTF-8, deberá asignarse a la propiedad del sistema "file.encoding" el valor "UTF-8", mediante la siguiente línea de comandos:

```
java -D -jar wipo-sequence-validator.jar
```

Se puede acceder a la API del Validador a través de Swagger UI:

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

Se puede acceder a la API del Validador en los siguientes puntos finales:

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

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

en los que la Oficina de PI deberá reemplazar:

- [host-name] por el nombre de anfitrión del servidor; y
- [version] por la versión de la API del Validador (por ejemplo, v1.0).

Por defecto, el servidor se ejecutará en el puerto 8080. Para modificarlo deberá añadirse la opción de línea de comandos "--server.port" , como se indica a continuación:

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

Por defecto, el Validador utilizará los ajustes de memoria predeterminados de la máquina virtual Java. El tamaño máximo de almacenamiento predeterminado para la asignación dinámica de memoria es una cuarta parte de la memoria física disponible.

Para modificar ese tamaño máximo, hay que utilizar la opción "-Xmx" cuando se ejecuta el Validador desde la línea de comandos[2]:

2.1.1. Implementación del Validador como una aplicación ejecutable

Asimismo, el Validador puede instalarse como un servicio gestionado por el sistema operativo de modo que, por ejemplo, se ejecute al arrancar el equipo.

Además, el archivo JAR de Spring Boot para instalarlo de ese modo podrá configurarse en cualquiera de los sistemas operativos con los que WIPO Sequence es compatible: Windows, Linux y macOS.

A continuación figura el enlace a una guía en la que se explica cómo crear en cada sistema operativo un servicio de sistema que ejecute un archivo JAR. También se proporciona información sobre cómo configurar las diferentes opciones del servicio y la ejecución de la aplicación:

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. Implementación del Validador como un archivo WAR de servicio web

Como segundo tipo de archivo binario, se puede instalar un paquete WAR en un servidor de aplicaciones Java, como Apache Tomcat 8.5.

Nota: se requiere un contenedor compatible con Servlet 3.1

Las instrucciones que figuran a continuación se refieren a un servidor de aplicaciones Tomcat. La variable "$TOMCAT_ROOT" corresponde al directorio raíz del servidor Tomcat, y su valor deberá sustituirse por el valor correspondiente a la ruta del archivo:

a) Detener el servidor: “$TOMCAT_ROOT\bin\catalina.bat stop”
b) Copiar el WAR en “$TOMCAT_ROOT\webapps\wipo-sequence-validator.war”
c) Iniciar el servidor: “$TOMCAT_ROOT\bin\catalina.bat start”

Nota: Para garantizar que Java permita abrir archivos en formato UTF-8, deberá asignarse a la propiedad del sistema “file.encoding” el valor “UTF-8” al iniciarse el servidor de aplicaciones, añadiendo lo siguiente en la línea de comandos: -D"file.encoding=UTF-8"

Se puede acceder a la API del Validador a través de Swagger UI, como ya se ha indicado:

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

Se puede acceder a la API del Validador en los siguientes puntos finales:

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

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

La Oficina de PI deberá reemplazar:

- [host-name] por el nombre de anfitrión del servidor; y
- [version] por la versión de la API del Validador (por ejemplo, v1.0).

Por defecto, el servidor se ejecutará en el puerto 8080. Para cambiar el puerto, hay que modificar el archivo de configuración de Tomcat según se indica en:

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

Por defecto, el Validador utilizará los ajustes de memoria predeterminados de la máquina virtual Java. El tamaño máximo de almacenamiento predeterminado para la asignación dinámica de memoria es una cuarta parte de la memoria física disponible.

Para modificar ese tamaño máximo, hay que utilizar la opción “-Xmx” cuando se ejecuta el Validador desde la línea de comandos, como se ha indicado en la sección 2.1.

3. Informe de verificación

La herramienta genera un informe de verificación en formato XML con la estructura que se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerificationReport productionDate="YYYY-MM-DD" sourceFileName="[ST.26 filename]">
   <VerificationMessageBag>
      <VerificationMessage>
         <Severity>[ERROR | WARN | XML_WARN |  XML_ERROR]</Severity>
         <DataElement>[ST.26 element]</DataElement>
         <DetectedSequence>[Sequence ID]</DetectedSequence>
         <DetectedValue>[value]</DetectedValue>
         <MessageKey>[Message key]</MessageKey>
         <ParameterBag>
                <Parameter key="param key">Param value</Parameter>
         <ParameterBag>
         <LocalizedMessage> [Localized message] </LocalizedMessage>
      </VerificationMessage>

...

   </VerificationMessageBag>
</VerificationReport>
```

En el Anexo I del presente manual se incluye un ejemplo de informe de verificación, junto con los valores permitidos para estos componentes contenidos en el Anexo III. En lo que atañe a la gravedad indicada, téngase en cuenta la siguiente categorización:

- ERROR – notificación de error durante la validación completa
- WARNING – notificación de advertencia durante la validación completa
- XML_ERROR – notificación de error durante la validación de forma
- XML_WARN – notificación de advertencia durante la validación de forma

4. Pedido de punto final de devolución de llamada

El pedido realizado al Validador por el punto final de devolución de llamada deberá contener los siguientes parámetros, que incluirán las ubicaciones de los archivos, así como información sobre el proceso de validación:


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


En el campo “seqlInputLocation” del pedido de validación se indicará la ruta de acceso al archivo XML de la lista de secuencias que se va a validar. Si la Oficina no rellena este campo, la herramienta intentará validar el archivo XML con el nombre “nameFile” ubicado en la carpeta predeterminada “Inbox”. El parámetro “nameFile” permite identificar el archivo de la lista de secuencias que se va a validar.

En el parámetro “verificationReportOutputPath” del pedido se proporcionará la ubicación del archivo XML del informe de verificación generado por la herramienta. Si se deja en blanco o se introduce una ruta de archivo no válida, el informe de verificación se guardará en la carpeta predeterminada “Reports”.

4.1. Formato de pedido de punto final de devolución de llamada

Si se configura la propiedad “api.URL”, el Validador intentará enviar los resultados de la validación a un punto final con el URL especificado.

La comunicación entre el Validador y el punto final de devolución de llamada deberá establecerse conforme al siguiente contrato de servicio web (definido en YAML):


```
openapi: 3.0.0
info:
  description: Callback for the WIPO Sequence Validator
  version: "1.0"
  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:
```

```yaml
          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
```

```yaml
        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
```

El pedido deberá ser una instancia JSON con la siguiente estructura:

```
"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"
}
```


A continuación figura un ejemplo de instancia JSON enviada al punto final externo que realizó la llamada al Validador:

4.2. Informe de verificación

Como se menciona en la subsección 3, tras la validación, el informe de verificación generado se guarda en la carpeta indicada en la ruta establecida en "verificationReportOutputPath", que en el ejemplo proporcionado es: "C:/temp/report.xml".

El contenido del informe se envía al punto final de devolución de llamada dentro del campo "errorSummary" de "ServiceRequest", como se muestra en los ejemplos de pedidos proporcionados en la subsección 4.

5. Configuración

5.1. Configuración predeterminada

La configuración del Validador se establece mediante un archivo de propiedades —el archivo application.properties—, que tiene por defecto los siguientes valores:

```
########### 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/

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

#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/

alternativeResourceBasePath=${app.basePath}alt_resources

#Preferences
#To enable the rule VXQV49 set this value to true, default value is false.
app.preferences.optionalEnglishQualifierValue=false
# Please enter either: ERROR or WARNING to specifiy the type of the verification me
ssage for the rule
 VXQV_49, default value is "WARNING".
app.preferences.optionalRuleType=WARNING

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

api.URL=
```

```
## -- Watcher

# These properties control the process looking for files in the folders to be proce
ssed
# (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
processing.enabled=true


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

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


# HEALTH ENDPOINT
management.endpoints.jmx.exposure.include=health
# Show details of health endpoint
management.endpoint.health.show-details=always
```

Para modificar el valor de los parámetros indicados se deberá generar otro archivo application.properties. En la documentación de Spring Boot se indican varias opciones: 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

Lo más sencillo es crear un nuevo archivo application.properties que se buscará automáticamente en las ubicaciones que figuran a continuación, en el orden de prioridad indicado:

a) La carpeta "/config" dentro del directorio en el que se ejecuta la aplicación *[Nota: si el Validador se implementa como un archivo WAR en Tomcat, esta carpeta estará dentro de la carpeta "lib"; por ejemplo, "/opt/apache-tomcat/lib/config"]*;

b) El directorio en el que se ejecuta la aplicación *[Nota: si el Validador se implementa como un archivo WAR en Tomcat, estará en la carpeta "lib"; por ejemplo, "/opt/apache-tomcat/lib/"];*

c) El paquete "/config" en el $classpath de la aplicación; o en

d) La raíz del $classpath de la aplicación

La ruta y el nombre del archivo de configuración también se pueden especificar cuando se inicie la herramienta mediante la línea de comandos:

- Si la implementación del Validador se realiza con un archivo JAR:

java -jar -Dspring.config.location= <PATH_TO_FILE> wipo-sequence-validator.jar

- Si la implementación del Validador se realiza con un archivo WAR en Tomcat, deberá añadirse la siguiente entrada a CATALINA_OPTS:

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

Cuando se utiliza un archivo WAR, también es posible copiar el nuevo archivo application.properties en la carpeta “WEB-INF/classes” de la aplicación web, o editarse el existente.

5.2. Configuración de la norma de verificación VXQV_49

```
app.preferences.optionalEnglishQualifierValue=false
app.preferences.optionalRuleType=WARNING
```

El valor `optionalEnglishQualifierValue` contenido en el archivo application.properties se puede configurar en modo 'true' si el usuario quiere activar la norma VXQV_49, y se puede configurar la gravedad de la norma actualizando el valor `optionalRuleType` a 'ERROR' o 'WARNING'. En la imagen anterior se muestran los valores por defecto de estas dos propiedades.

5.3. Comprobar la salud del punto final

```
# HEALTH ENDPOINT
management.endpoints.jmx.exposure.include=health
```

El Validador ha implantado un punto final /health que proporciona información básica sobre el estado de salud de la aplicación.

Para explorar el punto final /health , el URL es http://localhost:8080/wipo-sequence-validator/actuator/health. El punto final debería mostrar los siguientes estados:

- 'UP' siempre que la aplicación esté en buen estado.
- 'DOWN' si la función no funciona adecuadamente debido a cualquier problema, por ejemplo, sobre conectividad con la base de datos o falta de espacio en el disco, etcétera.

El punto final /health mostrará únicamente 'UP' o 'DOWN' en su estado. La siguiente propiedad en el archivo application.properties ofrece información completa, incluido el estado de cada indicador de salud verificado durante este procedimiento.

```
# Show details of health endpoint
management.endpoint.health.show-details=always
```

El punto final /health actualmente incluye la información completa sobre “DiskSpaceHealthIndicator”, que se ejecuta durante el procedimiento de chequeo.

El punto final relativo a la salud de la herramienta se visualizará del siguiente modo, como una serie de pares de valores clave, e incluirá la siguiente información (véase el ejemplo):

{"status":"UP","details":{"diskSpace":{"status":"UP","details":{"total":511123124224,"free":373225091072,"threshold":10485760}}}}

5.4. Mensajes localizados

El Validador puede proporcionar mensajes localizados, en particular en el informe de verificación, en cualquiera de los diez idiomas oficiales del PCT (español, alemán, árabe, chino, coreano, francés, inglés, japonés, portugués y ruso).

Por defecto, los mensajes se proporcionan en inglés. Para que el Validador suministre esos mensajes en otros idiomas, deberá configurarse el parámetro "validator_locale" del archivo application.properties con el código de idioma adecuado.

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

Nota: Para que se apliquen las propiedades establecidas en el nuevo archivo application.properties es necesario reiniciar el Validador.

5.5. Nombres de organismos personalizados

Las Oficinas pueden proporcionar los nombres de sus organismos personalizados, que no estén en la lista original predefinida de nombres de organismos, creando el archivo "custom_organism.json" con la lista de nombres en la carpeta "alternativeResourceBasePath". El archivo deberá tener la siguiente estructura:

Nota: Todos los organismos se incluirán en un único archivo JSON, en lugar de incluirse en distintos archivos JSON según la letra del alfabeto, como es el caso de los nombres de los organismos de la lista predefinida.

5.6. Establecimiento de la DTD prevista en la Norma ST.26 como referencia

Por defecto, el Validador utiliza como referencia la última versión de la DTD prevista en la Norma ST.26 de la OMPI. En concreto, la versión actual del Validador utiliza la versión 1.3 de la DTD[3].

En el repositorio de documentación del Validador, ubicado en la carpeta "/src/main/resources" del código fuente (ruta definida en el archivo JAR o WAR), se incluye una copia de la última versión de la DTD prevista en la Norma ST.26. Se establece como referencia en el archivo "catalog.xml", ubicado en esa misma carpeta, según se indica a continuación:

Más adelante se describe la manera de añadir otra DTD. La validación se realizará tomando como referencia la versión de la DTD establecida en la declaración DOCTYPE del archivo XML. El parámetro "publicId" servirá para identificar la ubicación del archivo DTD que se utilizará. Si "publicId" no está incluido en el catálogo XML, el sistema intentará localizar el archivo DTD en la carpeta raíz donde se está ejecutando el proceso Java.

5.6.1. Modificación de la versión de la DTD utilizada como referencia para la validación

Para poder validar archivos conforme a la Norma ST.26 de la OMPI utilizando como referencia una versión más antigua de la DTD prevista en la Norma, es preciso que el Validador tenga acceso al archivo DTD correspondiente a dicha versión.

Para ello, son posibles los siguientes dos enfoques:

[3] Válida a partir del 18 de abril de 2022.

a) Se deberá descomprimir el archivo JAR, incluir una copia del nuevo archivo DTD en la carpeta “src/main/resources”, y modificar el archivo “catalog.xml” añadiendo una nueva entrada para establecer como referencia la versión alternativa de la DTD de la Norma ST.26 o editando la entrada existente.

Ejemplo:

b) En lugar de modificar el archivo JAR, se pueden seguir los siguientes pasos:

   i) Se copiarán en una carpeta local el archivo “catalog.xml” y todos los archivos DTD;
   ii) Se modificará el archivo “catalog.xml” para establecer como referencia la versión alternativa de la DTD de la Norma ST.26; y
   iii) En la ejecución del proceso Java, se deberá configurar la siguiente propiedad del sistema según se indica: “xml.catalog.files=<path_to_catalog.xml>”.

Nota: En Tomcat para Windows se añadiría esta variable de entorno:

[IMPORTANTE: Cuando se modifique la versión de la DTD de la Norma ST.26, se podrá realizar la validación de forma del archivo XML utilizando la DTD establecida como referencia. Sin embargo, la validación completa requerirá probablemente que se modifique el código fuente para que se verifiquen las normas pertinentes. Por consiguiente, se recomienda utilizar varias DTD solo cuando se lleve a cabo la validación de forma.]

6. API REST del Validador

En esta sección se especifican los tres servicios que ofrece la API del Validador:

- Validar un archivo ubicado en la carpeta “Inbox”;
- validar un archivo como parte del pedido; y
- solicitar información acerca del estado de una validación.

En el Anexo II se proporciona la especificación completa de la API para el tercer servicio (especificación completa OpenAPI 3.0 [en YAML]).

6.1. Validación de archivos conforme a la Norma ST.26 de la OMPI

Asignación de pedido	/api/v1/validate
Método	POST
Anotación @Consumes	application/json
Anotación @Produces	application/json
Operación	Solicita la validación de un archivo ubicado en la carpeta “Inbox” para verificar su conformidad con la Norma ST.26 de la OMPI. Devuelve un único valor de “verificationID” para indicar el estado en que se encuentra el pedido de validación.

Pedido	{
	"currentApplicationNumber": "string",
	"currentSEQLVersionNumber": "string",
	"parentApplicationNumber": "string",
	"parentSEQLVersionNumber": "string",
	"seqlInputLocation": "string" (Location of Input.xml file),
	"verificationReportOutputPath": "string" (Destination of report.xml),
	“nameFile”: "string",
	“type”: "string" (Possible values: full | formality),
	}

Respuestas	• '202': “Accepted” (“Pedido aceptado”). El archivo ha pasado la validación de forma y se ha iniciado la verificación de las normas operativas conforme a la Norma ST.26 de la OMPI. Este código de estado irá acompañado de un mensaje de respuesta con un código único (“verificationID”) que indicará el estado del proceso de verificación. El archivo que se va a validar conforme a la Norma ST.26 de la OMPI se mueve a la carpeta “Process” durante el proceso.

	• '400': "Bad request" ("Pedido incorrecto"). El pedido REST no se formuló correctamente o el archivo no pasó la validación simple en XML. Este código de estado irá acompañado de un mensaje de respuesta con información sobre el error. • '404': "Not Found" ("No encontrado"). El archivo que se va a validar conforme a la Norma ST.26 de la OMPI no se encontró en la carpeta "Inbox". • '500': "Internal Server Error" ("Error interno del servidor"). Se ha producido un error interno. Este código de estado irá acompañado de un mensaje de respuesta con información sobre el error.
Precondición	El archivo que se va a validar conforme a la Norma ST.26 de la OMPI deberá guardarse en la carpeta "Inbox".
Postcondición	El archivo validado conforme a la Norma ST.26 de la OMPI se moverá a la ubicación "verificationReportOutputPath" o a la carpeta "Outbox", con la siguiente ubicación: "/[outbox]/[verificationID]/[file.xml]" El informe de verificación se genera en: "/[reports]/ [verificationID]/report_[file.xml]"

Especificación OpenApi 3.0 correspondiente [en 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 indicating a unique
 code for

retrieving the verification report (verificationID). ST26 file is moved to the �pr
ocess� 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 in
dicating 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
```

6.2. Pedido de información acerca del estado de la validación

Asignación de pedido	/api/v1/status
Método	POST
Anotación @Consumes	application/json
Anotación @Produces	application/json
Operación	Solicita información acerca del estado de la validación de un archivo específico conforme a la Norma ST.26 de la OMPI.
Pedido	{ verificationID: {type: string} }
Respuestas	• '200': “Success” (“Pedido correcto”). Este código de estado irá acompañado de un mensaje de respuesta con uno de los siguientes códigos que indicará el estado del proceso de verificación: • “RUNNING”: El archivo se está procesando. • “FINISHED-VALID”: El archivo ha pasado correctamente la validación de forma. El resultado de la validación puede consultarse en la carpeta “Reports”. • “FINISHED-INVALID”: El proceso se ha completado pero el archivo no ha pasado la validación de forma. El resultado de la validación puede consultarse en la carpeta “Reports”. • “NOT_FOUND”: El parámetro "verificationID" no se ha encontrado. • “VERIFICATION_ID_ERROR”: El parámetro "verificationID" no se ha incluido en el pedido. • ‘400’: “Bad request” (“Pedido incorrecto”). El pedido REST no se formuló correctamente. • 500': “Internal Server Error” (“Error interno del servidor”). Se ha producido un error interno. Este código de estado irá acompañado de un mensaje de respuesta con información sobre el error.
Postcondición	El Validador proporciona el estado de la validación.
Hipótesis	-

Especificación OpenApi 3.0 correspondiente [en YAML]

```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 avail
able
          in the reports folder) FINISHED-INVALID

(the file passed the formality validation and the result of the validation is avail
able
          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
```


[Sigue el Anexo I]

Anexo I: Ejemplo de informe de verificación


```xml
<?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.</LocalizedMes
sage>
        </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 inventio
n 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 inventio
n title must
            be entered.</LocalizedMessage>
        </VerificationMessage>
    </VerificationMessageBag>
</VerificationReport>
```


[Sigue el Anexo II]

Anexo II: Especificación completa de la API (en YAML)


```yaml
swagger: '2.0'
info:
  description: 'API for the WIPO Sequence Validator'
  version: '1.0'
  title: 'WIPO Sequence Validator API'
host: 'localhost:8080'
basePath: /
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
      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) FINISH
ED-VALID

(the file passed the formality validation and the result of the validation is avail
able
          in the reports folder) FINISHED-INVALID

(the file passed the formality validation and the result of the validation is avail
able
           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
/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 indicating a unique
 code for
      retrieving the verification report (verificationID). ST26 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 in
dicating
      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
definitions:
  ValidationRequest:
    type: object
    required:
      - nameFile
      - type
    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 listing'
      currentSEQLVersionNumber:
```

```
        type: string
        example: 1.2
        description: 'the version number of this sequence listing (internally assig
ned 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 generated
 by the tool.'

    title: ValidationRequest
    description: 'Class representing a response Validation status File by the appli
cation.'
  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 applica
tion.'
  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: 'Validation 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
.'
```


[Sigue el Anexo III]

Anexo III: Nombres de las propiedades (en JSON)

```json
{
    "dtdVersion": "PROPERTY_NAMES.DTD_VERSION",
    "fileName": "PROPERTY_NAMES.FILE_NAME",
    "softwareName": "PROPERTY_NAMES.SW_NAME",
    "softwareVersion": "PROPERTY_NAMES.SW_VERSION",
    "productionDate": "PROPERTY_NAMES.PRODUCTION_DATE",
    "project": "PROPERTY_NAMES.PROJECT",
    "nonEnglishFreeTextLanguageCode": "PROPERTY_NAMES.NON_ENGLISH_FREE_TEXT_LANGUAGE_CODE",
    "originalFreeTextLanguageCode": "PROPERTY_NAMES.ORIGINAL_FREE_TEXT_LANGUAGE_CODE",
    "name": "PROPERTY_NAMES.PROJECT_NAME",
    "applicationIdentification": "PROPERTY_NAMES.APPLICANT_IDENTIFICATION",
    "applicationIdentification.filingDate": "PROPERTY_NAMES.APPLICANT_IDENTIFICATION",
    "applicantFileReference": "PROPERTY_NAMES.APPLICANT_FILE_REFERENCE",
    "priorityInformationBag": "PROPERTY_NAMES.PRIORITY_INFORMATION_BAG",
    "earliestPriorityApplicationIdentification": "PROPERTY_NAMES.EARLIEST_PRIORITY_APPLICATION",
    "earliestPriorityApplicationIdentification.filingDate": "PROPERTY_NAMES.EARLIEST_PRIORITY_APPLICATION",
    "applicantName": "PROPERTY_NAMES.APPLICANT",
    "applicantList": "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",
    "inventorList": "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",
    "sequenceTotalQuantity": "PROPERTY_NAMES.SEQUENCE_TOTAL_QUANTITY",
    "sequenceDataBag": "PROPERTY_NAMES.SEQUENCE_DATA_BAG",
    "sequenceIDNumber": "PROPERTY_NAMES.SEQUENCE_ID_NUMBER",
    "length": "PROPERTY_NAMES.SEQUENCE_LENGTH",
    "INSDSeqMoltype": "PROPERTY_NAMES.SEQ_MOL_TYPE",
    "INSDQualifierMolType": "PROPERTY_NAMES.QUAL_MOL_TYPE",
    "organism": "PROPERTY_NAMES.ORGANISM",
    "division": "PROPERTY_NAMES.DIVISION",
    "INSDSeq_other-seqids": "PROPERTY_NAMES.INSDSEQ_OTHER-SEQIDS",
    "featureTable": "PROPERTY_NAMES.FEATURE_TABLE",
    "featureKey": "PROPERTY_NAMES.FEATURE_KEY",
```

```
    "featureLocation": "PROPERTY_NAMES.FEATURE_LOCATION",
    "featureQuals": "PROPERTY_NAMES.FEATURE_QUALS",
    "qualifierId": "PROPERTY_NAMES.QUALIFIER_ID",
    "qualifierName": "PROPERTY_NAMES.QUALIFIER_NAME",
    "qualifierValue": "PROPERTY_NAMES.QUALIFIER_VALUE",
    "qualifierTranslatedValue": "PROPERTY_NAMES.QUALIFIER_TRANSLATED_VALUE",
    "INSDSeqSequence": "PROPERTY_NAMES.SEQ_SEQUENCE"
}
```

Debe tenerse presente que esta información es parte del código fuente y se actualiza según sea necesario al momento de la aplicación.

[Fin del documento]