WIPO Sequence Validator – Operation Manual

Version 1.1.0

The purpose of this document is to support Intellectual Property Offices to deploy the WIPO Sequence Validator microservice and also to support configuration of the Validator.

Table of contents

1. Introduction

The main purpose of the WIPO Sequence Validator (hereinafter referred to as 'Validator' or 'the tool') is to provide IP Offices (IPOs) with a microservice to validate XML files in WIPO ST.26 format to ensure they are compliant with WIPO Standard ST.26. Although a sequence listing which has been drafted using the WIPO Sequence desktop application will be WIPO ST.26-compliant, users may use whatever tool they deem most suitable.

The objective of this document is to explain the build, deployment, settings and file consumer system of the tool that is detailed in the following sections.

1.1. Validator Workflow Overview

The tool provides the following four use cases:

- Validation of an WIPO ST.26 file;
- Request the status of a running validation;
- Update configuration files (only the IPO Admin); and
- Call a callback endpoint with the result of the validation process once the process is completed. *Note: this callback endpoint[1] is outside the scope of the Validator. It is up to the Offices developing and setting up this service to establish the endpoint.*

The tool is composed of a JAR file that can be executed as a web service or a WAR file that can be deployed on a Tomcat server.

In both cases, for validation of the WIPO ST.26 sequence listing, the tool consumes files from a local file system and generates a verification report with validation results and, optionally, returns the results of the validation process, i.e., verification report, by calling a callback endpoint.

The main workflow of the Validator is as follows:

(i) The respective IPO IT system saves a WIPO ST.26 XML file in a default "Inbox" folder or the specified one within the request.

(ii) The IPO system initiates a HTTP Post requesting the validation of the file. Depending on the configuration, the IPO system can request a "full" or a "formality" validation of the file. The "formality" validation process will check whether the ST.26 file is an XML file and validates the file against the ST.26 DTD. The "full" validation process will validate the ST.26 file against the business verification rules, derived from the content of ST.26 as well as conducting the "formality" validation process.

Note: It is recommended to use the "formality" validation process for online filing acceptance system only as this can be performed synchronously while the "full" validation is recommended for batch processing as it will take much longer.

[1] A callback endpoint here is a unique address identified by a URI to which request messages can be sent

(iii) Once the validation is complete, a response will be provided indicating whether or not the file has passed the "formality" validation and, in the instance where the IPO IT system has selected "full" validation, additionally whether or not the business rule validation process has started correctly.

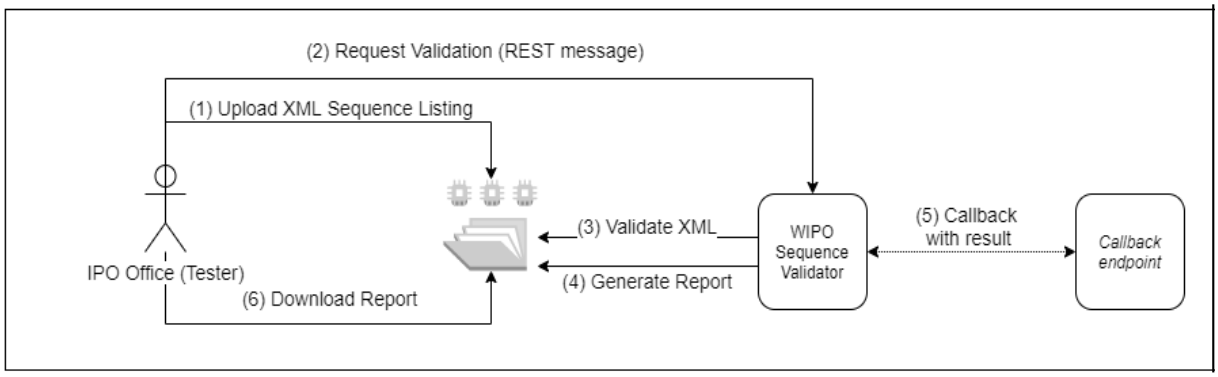
(iv) If the Validator is conducting a "full" validation, it retrieves the XML file from the "Inbox" folder and starts the business rule validation process, and then conducts the following:

(v) The Validator generates an XML report file ("Verification Report") in a specified "Output" folder and moves the validated WIPO ST.26 XML file to an "Outbox" folder.

(vi) After the business rule validation process has completed, the callback endpoint, if configured, is called from the Validator, and the request is populated with additional information related to the validation process. The structure of the request and some sample data is provided in section 2.2.2 below.

(vii) The callback endpoint should either return an empty or a success code in the response (no errors). *[Note: that this step is only executed if the external web service has been made available and the call has been configured in the Validator.]* Connectivity between the Validator and the callback endpoint is also required. As mentioned above, the external web service does not form part of the Validator and it should be developed and configured by Offices according to the contract defined below.

(viii) The IPO system can retrieve the Verification Report from the "Report" folder.



Note: the WIPO Sequence Validator complies with the WIPO Standard for processing and communicating intellectual property data using Web APIs: WIPO ST.90.

1.2. Definition of the Validator file system structure

The structure of the file system used by the Validator consists of five folders:

- **"Inbox" folder**: This is the local folder where WIPO ST.26 files are provided by an IPO for validation.

- **“Process” folder**: This is the local folder through which the files in the “Inbox” temporarily pass during processing. This folder contains two subfolders
    - **“Full validation” folder**: stores the files pending a full validation
    - **“Formality validation” folder**: stores the files pending a formality validation
- **“Outbox” folder**: Once the validation is complete, the application stores the source of the WIPO ST.26 file in this local folder.
- **“Report” folder**: This is the local folder where the results of the validation are saved in a verification report file.
- **“Params” folder**: This the local folder where a JSON (.json) file with all the validation parameters from the validation request are located in order to provide parameters for the asynchronous deep validation process.

An example file system structure is provided below:

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

2. Deployment of the WIPO Sequence Validator

As indicated previously, the Validator is provided as one of two binary formats, listed below. Depending on the type of infrastructure in which the Office wants to deploy the Validator, the IPO will have a preference for selecting one type of binary over the other.

The two binaries that the Validator are provided as are:

- **Binary SpringBoot JAR**: This binary is an executable JAR file. This requires Java 8 to be installed.
- **War Package Binary:** This binary is intended to be deployed on a Servlet container. An application server compatible with Spring Boot 2 and Servlet Spec 3.1+ is required, such as Tomcat 8.5.

The following sections detail the deployment of the Validator as a Spring Boot app or as a WAR within a Java Application Server.

2.1. Starting the Validator as a Spring Boot JAR

The Spring Boot JAR contains an embedded server, which allows the deployment of the Validator API without requiring a separate server. This greatly simplifies the configuration and deployment at the infrastructure level.

In order to run the embedded server, the following command should be executed.

Note: Java 8 must already be installed on the server: Since Java does not guarantee the use of UTF-8, the system property "file.encoding" must be set to "UTF-8". This can be done by including the following:

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

The Validator API can be accessed through a Swagger UI:

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

The Validator API is accessible in the following endpoints:

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

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

where the IPO must make the following changes:

- [host-name] must be replaced by the server host name; and
- [version] should be replaced by the version of the Validator API (e.g., v1.0).

By default the server will run on port 8080, to change the port the "--server.port" command line option should be added as shown here:

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

By default the Validator will use the Java Virtual Machine (JVM) default memory settings. The default maximum heap size is one fourth of the physical memory available.

In order to modify the maximum heap size, the "-Xmx" option must be used when executing using the command line[2]:

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

2.1.1. Deploying Validator as an executable application

The Validator can also be installed as a service managed by the operating system to support its execution with the start-up of the operating system, for instance.

It is possible to configure a Spring Boot JAR file in this manner for all platforms supported by WIPO Sequence: Windows, Linux and Mac OS.

The following guide provides detail how to create a system service that executes the JAR file for each operating system. It also provides information on how to configure the different options of the service and the execution of the 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. Deployment as a WAR web service

For the second type of binary provided, the WAR package can be deployed in an existing Java application server such as Apache Tomcat 8.5.

Note: a container compatible with Servlet 3.1 is required.

The following instructions are for a Tomcat application server. Here, "$TOMCAT_ROOT" refers to the root folder of the Tomcat server, and this value should be replaced with the relevant value for the file path:

(a) Stop server: "$TOMCAT_ROOT\bin\catalina.bat stop"

(b) Copy WAR to "$TOMCAT_ROOT\webapps\wipo-sequence-validator.war"

(c) Start server: "$TOMCAT_ROOT\bin\catalina.bat start"

Note: since Java does not guarantee the use of UTF-8, the system property "file.encoding" must be set to "UTF-8" in the start-up of the application server. This can be done by including the following: -D"file.encoding=UTF-8"

The Validator API can be accessed through a Swagger UI, as indicated above:

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

The Validator API is accessible in the following endpoints:

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

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

where the IPO must make the following changes:

- [host-name] must be replaced by the server host name, and
- [version] should be replaced by the version of the API (e.g., v1.0).

By default, the server will run on port 8080. To change this over to another port the Tomcat configuration file should be modified by following the instructions provided here:

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

By default, the Validator will use the JVM default memory settings. The default maximum heap size is one fourth of the physical memory available.

In order to modify the maximum heap size, the "-Xmx" option must be used when executing using the command line, as indicated above in Section 2.1.

2.2.1. Verification Report

The verification report generated by the tool is in XML format and the template used is provided below:

```
<?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>
```

An example of this verification report is provided as Annex I to this manual with allowable values for tag at Annex III.

2.2.2. Callback Endpoint request

The request made by the callback endpoint to the Validator should include the following parameters, detailing the file locations and the validation process:

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

The “seqlInputLocation” field of the validation request should be set to indicate the path of the XML sequence listing to be validated. If the Office leaves this field empty, the tool will try to validate the XML file with the filename “nameFile” located at the default “Inbox” folder. The “nameFile” parameter identifies the sequence listing file to be validated.

The “verificationReportOutputPath” within the request will provide the file location of the verification report (.xml) generated by the tool. If the user leaves the field in blank or introduces an invalid file path, the verification report will be stored in the default “Reports” folder.

2.2.2.1 Callback Endpoint Request Format

If the property “api.URL” is configured, the Validator will try to send the results of the validation to an endpoint with the URL specified.

In order to communicate with the Validator, the callback endpoint must comply with the following web service contract (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
```

Furthermore, the request should be a JSON object with this structure:

```
{
 "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"
}
```

This is an example JSON object that will be sent to the external endpoint which made the call to the Validator:


```
{
   "processID": "1608194222169dWE",
   "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. Verification Report

As mentioned in section 2.2.1, after validation, the verification report generated is located at the "verificationReportOutputPath", which in the example provided is at: "C:/temp/report.xml".

The content of this report is sent to the callback endpoint within the "errorSummary" field of the “ServiceRequest”. An example of this field is provided in the example requests provided above in section 2.2.2.

2.3. Configuration

2.3.1. Default settings

The Validator is configured by using a properties file. The default “application.properties” file has the following values:

```
############ 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
```

In order to modify the value of the parameters provided here an alternative "application.properties" file should be used. There are several options as detailed in the Spring Boot documentation: 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

The simplest option would be to provide a new "application.properties" file that will be searched for in the following locations, by order of operation:

(a) A "/config" folder within the current directory *[Note: if the Validator is deployed as a WAR file in Tomcat, this folder would be under "lib" folder, e.g., "/opt/apache-tomcat/lib/config"]*;

(b) The current directory *[Note: if the Validator is deployed as a WAR file in Tomcat, this folder would be under "lib" folder, e.g., "/opt/apache-tomcat/lib/"]*;

(c) A $classpath or config package; then

(d) The $classpath root.

Also, the path and name of the configuration file can be specified by establishing the parameter on the command line when starting the tool:

(a) For a JAR deployment:

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

(b) For a WAR deployment on Tomcat, the following entry to CATALINA_OPTS should be added:

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

When using the WAR deployment, the new "application.properties" file can also be copied into the "WEB-INF/classes" folder of the web application.

Note: The Validator should be restarted in order for the properties set out in the new "application.properties" file to be applied.

2.3.2. Localized messages

The Validator can provide a localized message, for instance in the Verification Report, in each of the official ten PCT languages (Arabic, Chinese, English, French, German, Portuguese, Japanese, Korean, Russian and Spanish). By default, these messages are provided in English. In order to configure the Validator to provide these messages in other language, the "validator_locale" parameter of the "application.properties" file must be set to the appropriate language code. For instance, "validator_locale=es".

2.3.3. Custom Organism Names

In order for Offices to provide their own custom organism names, which do not form part of the original pre-defined list of organism names, a list of custom organisms can be provided by creating a new file, called "custom_organism.json", in the "alternativeResourceBasePath" folder. This file should have the following structure:

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

Note: unlike the pre-defined list of organism names, all organisms are contained within a single JSON file, rather than be separated out into a JSON file for each letter of the alphabet.

2.3.4. Referencing ST.26 DTD files

By default, the Validator references the latest version of ST.26 DTD. The current version of WIPO Sequence Validator is based on version 1.3 of WIPO ST.26 DTD[3] .

This copy of the latest version of the ST.26 DTD is included in the Validator library located in the "/src/main/resources" folder of the source code (this is the defined file path referenced within the JAR or WAR file). It is referenced in the "catalog.xml" file in the same folder, as shown below:

```
<?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>
```

Instructions on how to include a new DTD are detailed below. During the validation, the version of the DTD set in the DOCTYPE declaration of the XML file will be used. Firstly, the "publicId" will be used to identify the location of the DTD file to be used. If the "publicId" is not included in the catalog, the system will try to locate the DTD file in the root folder where the Java process is being executed.

How to indicate an alternative DTD version for the validation

In order to be able to validate WIPO ST.26 files that reference an older version of the ST.26 DTD, this ST.26 DTD file must be made available to the Validator to allow the appropriate validation.

In order to achieve this, there are two alternative approaches:

(a) Uncompress the JAR file and include a reference to the additional or alternative ST.26 DTD file in the "src/main/resources" folder. Modify the "catalog.xml" file and add a new entry for the additional ST.26 DTD or edit the existing one.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
```

[3] Valid as of January 14, 2021

```
<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) Instead of modifying the JAR file, the following steps should be followed:

(i) Copy “catalog.xml” and all the DTDs into a local folder;
(ii) Modify “catalog.xml” to include a reference to the additional ST.26 DTD; and
(iii) Set this Java system property on launch: “xml.catalog.files=<path_to_catalog.xml>”

Note: In Tomcat for Windows can be done by adding this environment variable:

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

[IMPORTANT: the inclusion of a different version of the ST.26 DTD will allow the ‘formality’ validation of the XML file against the referenced ST.26 DTD but the ‘full’ validation will likely require a change to the source code in order to implement the verification rules. Therefore it is recommended that the use of multiple DTDs only be used when conducting a ‘formality’ validation.]

3. Validator REST API

In this section, the use cases of the Validator API are specified. There are three services or use cases:

(a) Validate a file in the “Inbox” folder;

(b) Validate a file as part of the request; and

(c) Request the status of a validation.

The API specification for this (OAS 3.0 Complete API [YAML File]) service is provided in full as Annex II.

3.1. Validate WIPO ST.26 file

Request Mapping	/api/v1/validate
Method	POST
Consumes	application/json
Produces	application/json
Operation	Request the validation of an existing WIPO ST.26 file in the “Inbox” folder. Returns a unique “verificationID” for retrieving the status of the validation request

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

Responses	• '202': “Accepted”. The WIPO ST.26 file passed the formality validation and the business rule verification has started. This code will also include a response message indicating a unique code for retrieving the verification report (“verificationID”). The WIPO ST.26 file is moved to the “Process” folder for processing.

	• '400': "Bad request". The REST request was not well formed or the WIPO ST.26 file did not pass the XML validation. This code will be complemented with a response message indicating the detail of the error • '404': "Not Found". The WIPO ST.26 file was not found in the "Inbox" Folder • '500': "Internal Server Error". An internal error occurred. This code will be complemented with a response message indicating the detail of the error
Precondition	The WIPO ST.26 file must be provided in the "Inbox" folder, specified by the user.
Postcondition	The WIPO ST.26 file is moved to the location provided in verificationReportOutputPath or to the "Outbox" folder at the following location: "/[outbox]/[verificationID]/[file.xml]" The verification report is generated at: "/[reports]/ [verificationID]/report_[file.xml]"

Corresponding OAS 3.0 Definition [in YAML specification]

```
 /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. Request Validation Status

Request Mapping	/api/v1/status
Method	POST
Consumes	application/json
Produces	application/json
Operation	Request the validation status for a specific WIPO ST.26 File

Request	{
	verificationID: {type: string}
	}

Responses	- '200': Success. This code will also include a response message indicating a unique ID with the status of the verification process, selected from: o "RUNNING": the file is being processed o "FINISHED-VALID": the file successfully passed the formality validation and the result of the validation is available in the reports folder o "FINISHED-INVALID": the process is complete but the file did not pass the formality validation. The result of the validation is available in the reports folder o "NOT_FOUND": the "verificationID" has not been found o "VERIFICATION_ID_ERROR": the "verificationID" has not been included in the request - '400': Bad request. The REST request was not well formed - '500': Server Error. An internal error happened. This code will also include a response message indicating the detail of the error
Postcondition	The Validator provides status of validation.
Assumptions	-

Corresponding OAS 3.0 Definition [in YAML specification]

```
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
```

[Annex I follows]

Annex I: Example Verification Report

```
<?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>
```

[Annex II follows]

Annex II: Complete API specification (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.
```

[Annex III follows]

Annex III: Property Names (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",

}
```

[End of document]