# Table of Contents

1. **Introduction** ................................................................................................................................................... 5  
   1.1 Purpose of the Document ..................................................................................................................... 5  
   1.2 Intended Audience ................................................................................................................................ 5  
   1.3 Structure of the Document .................................................................................................................... 5  

2. **REST API description** ...................................................................................................................................... 6  
   2.1 Accessing the API ................................................................................................................................... 6  
   2.2 Security .................................................................................................................................................. 6  
      2.2.1 Authentication .............................................................................................................................. 6  
   2.3 Operations ............................................................................................................................................. 7  
      2.3.1 Submit Notice ............................................................................................................................... 7  
      2.3.2 Get Notice Information ................................................................................................................. 8  
      2.3.3 Render Notice ............................................................................................................................. 10  
      2.3.4 Search Notice .............................................................................................................................. 13  
   2.4 Objects ................................................................................................................................................. 15  
      2.4.1 notice_information ..................................................................................................................... 15  
      2.4.2 publication_info .......................................................................................................................... 18  
      2.4.3 ted_links ...................................................................................................................................... 18  
      2.4.4 validation_report ........................................................................................................................ 18  
      2.4.5 validation_report_items ............................................................................................................. 18  
      2.4.6 simple_result .............................................................................................................................. 18  
      2.4.7 page_result ................................................................................................................................... 19  
   2.5 Notice life cycle .................................................................................................................................... 20  
   2.6 Java REST client .................................................................................................................................... 20  
      2.6.1 Create an eSentool REST Client instance .................................................................................... 20  
      2.6.2 Call one of the REST operation ................................................................................................... 20
List of Tables

Table 1: Reference Documents ................................................................. Error! Bookmark not defined.
Table 2: Applicable Documents ................................................................. Error! Bookmark not defined.
Table 3: Abbreviations and Acronyms ....................................................... 4
Table 4: Definitions ...................................................................................... 4
Table 5: API credentials examples .............................................................. 6

List of Figures

Figure 1: Notice life cycle ........................................................................... 20
Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HyperText Transfer Protocol Secure</td>
</tr>
<tr>
<td>OP</td>
<td>Office des Publications Officielles des Communautés Européennes</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
</tbody>
</table>

*Table 1: Abbreviations and Acronyms*

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURL</td>
<td>A command line tool for getting or sending files using URL syntax.</td>
</tr>
</tbody>
</table>

*Table 2: Definitions*
1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The purpose of this document is to provide the required information to operate the eSentool system.

1.2 INTENDED AUDIENCE

The present document is intended to be read by the following people:

- OP IT Project Manager;
- OP Infrastructure Team.

1.3 STRUCTURE OF THE DOCUMENT

The document is organized as follows:

- Chapter 1 - Introduction: summarises the purpose and scope of this document;
- Chapter 2 - REST API description: describes the REST API of the eSentool application.


2 REST API DESCRIPTION

The eSentool application provides a REST API based on JSON to other applications in order to expose its services.

The eSentool application will expose the following REST operations:

- Submit Notice;
- Get Notice Information;
- Render Notice;
- Search Notice;

2.1 ACCESSING THE API

Two environments are available:

- Qualification: This environment should be used for test purpose and in order to get qualified. The qualification REST API is accessible at the following URL: https://esentool.ted.europa.eu/api/qualification/latest/.
- Production: This environment should be used to send notices for publication on TED. The production REST API is accessible at the following URL: https://esentool.ted.europa.eu/api/production/latest/.

Both environments contain the same operations.

If a resource from a different environment than the previously stated ones is requested, it will be the web application, and not the REST API, that will handle the request, thus generating a HTML response.

2.2 SECURITY

An authentication is required to access eSentool REST API due to data access restriction. An eSender can access only to the notice information of its own notices, and notices of its customers. However an eSender cannot render the notices of one of its customer. A customer can only access to its own notices.

The credentials to access to the eSentool REST API are the <eSender login>:<eSender web service password>. For the customers the credentials are <eSender login><customer login>:<eSender web service password>.

Examples:

<table>
<thead>
<tr>
<th>Company</th>
<th>Credentials</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSender TED123</td>
<td>TED123:TED123password</td>
</tr>
<tr>
<td>Customer customer1 of eSender TED123</td>
<td>TED123customer1:TED123password</td>
</tr>
<tr>
<td>Customer customer1 of eSender TED456</td>
<td>TED456customer1:TED456password</td>
</tr>
</tbody>
</table>

Table 3: API credentials examples

The eSender web service password could be edited in the eSentool web application.

2.2.1 AUTHENTICATION

To access to the eSentool REST API clients must be authenticated by using standard HTTP basic authentication over HTTPS.
2.2.1.1 Simple curl example

```
```

2.2.1.2 Supplying Basic Auth headers

1. Build a string of the form username:password
2. The created string should be Base64 encoded
3. Supply an "Authorization" header with content "Basic" followed by the encoded string. For example, the string "TED123:password" encodes to "VEVEMTIzOnBhc3N3b3Jk" in base64, so you would make the request as follows:

```
```

2.3 OPERATIONS

2.3.1 SUBMIT NOTICE

2.3.1.1 URI

POST /notice/submit

2.3.1.1.1 Request

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notice</td>
<td>formData</td>
<td>string</td>
<td>XML file base64 encoded</td>
</tr>
</tbody>
</table>

2.3.1.2 Response

2.3.1.2.1 200

In case of successful response a **notice_information** object is returned.

Example:

```
{
   "submission_id": "Q20150617-5001",
   "received_at": "2015-06-17T08:10:04Z",
   "status": "RECEIVED",
   "reason_code": null,
   "status_updated_at": "2015-06-17T08:10:04Z",
   "no_doc_ext": null,
   "form": null,
   "languages": [],
   "publication_info": null,
   "technical_validation_report": null,
   "validation_rules_report": null,
   "quality_control_report": null
}
```

2.3.1.2.2 400
2.3.1.2.2.1 In case the received notice is not valid Base64 scheme.

Example:

```json
{
  "timestamp": "2015-06-17T08:12:45Z",
  "status": 400,
  "error": "Bad Request",
  "exception": "The input is not in valid Base64 scheme",
  "message": "Invalid argument"
}
```

2.3.1.2.2.2 In case for other generic error.

Example:

```json
{
  "timestamp": "2016-05-25T18:24:55Z",
  "status": 400,
  "error": "Bad Request",
  "message": "This error might be caused by a misuse of the API. Please check parameters and API usage according to technical specification",
  "path": "/api/production//v1.0/notice/submit",
  "error_id": "651dfcdc-efef-47fc-9452-4ad16af9d03"
}
```

2.3.1.2.3 406

In case the Accept header is provided and set with a value different than ‘application/json’.

Example:

```json
{
  "timestamp": "2017-01-10T10:03:30Z",
  "status": "406",
  "error": "Not Acceptable",
  "message": "Not acceptable value for ‘Accept’ header. Only ‘application/json’ format is supported"
}
```

2.3.2 GET NOTICE INFORMATION

2.3.2.1 URI

GET /notice/{submission_id}

2.3.2.1.1 Request

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>submission_id</td>
<td>Path</td>
<td>string</td>
<td>The submission ID of the notice</td>
</tr>
</tbody>
</table>
2.3.2.2 Response

2.3.2.2.1 200

In case of successful response a notice_information object is returned.

Example:

```json
{
    "submission_id": "Q20150616-5075",
    "received_at": "2015-06-16T10:47:05Z",
    "status": "VALIDATION_ACCEPTED",
    "reason_code": null,
    "status_updated_at": "2015-06-16T10:47:05Z",
    "no_doc_ext": "2014-206001",
    "form": "F01",
    "languages": ["EN"],
    "publication_info": null,
    "technical_validation_report": {
        "type": "TECH",
        "items": [
            {
                "name": "T001",
                "valid": true,
                "severity": null,
                "message": "Xml is not valid against XSD",
                "details": null
            }
        ]
    },
    "validation_rules_report": {
        "type": "VALIDATION_RULES",
        "items": [
            {
                "name": "R006",
                "valid": true,
                "severity": null,
                "message": "Check that the XML file/notice contains only utf-8 characters",
                "details": null
            }
        ]
    },
    "quality_control_report": null,
    "ref_submission_id": null
}
```
2.3.2.2 400

In case for other generic error

Example:

```json
{
	"timestamp": "2016-05-25T18:24:55Z",
	"status": 400,
	"error": "Bad Request",
	"message": "This error might be caused by a miss use of the API. Please check parameters and API usage according to technical specification",
	"path": "/api/production//v1.0/notice/submit",
	"error_id": "651dfcdc-efef-47fc-9452-4ad16a9d03"
}
```

2.3.2.3 404

In case the related notice could not be found.

Example:

```json
{
	"timestamp": "2015-06-17T08:16:14Z",
	"status": 404,
	"error": "Not Found",
	"message": "Notice not found"
}
```

2.3.2.4 406

In case the Accept header is provided and set with a value different than ‘application/json’.

Example:

```json
{
	"timestamp": "2017-01-10T10:03;30Z",
	"status": 406,
	"error": "Not Acceptable",
	"message": "Not acceptable value for 'Accept' header. Only 'application/json' format is supported"
}
```

2.3.3 RENDER NOTICE

This operation allows rendering a notice in different formats: PDF, HTML, or regulation. PDF and HTML render the notice as displayed on TED website, while REGULATION render the notice as filled on eNotices.

The notice could be provided with the notice parameter, or by specifying the submission_id of an existing notice in eSentool.
Additionally the language of the rendering could be specified.

### 2.3.3.1 URI

POST /notice/render

#### 2.3.3.1 Request

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notice</td>
<td>formData</td>
<td>string</td>
<td>XML file base64 encoded</td>
</tr>
<tr>
<td>submission_id</td>
<td>formData</td>
<td>string</td>
<td>The submission ID of the notice to be rendered</td>
</tr>
<tr>
<td>format</td>
<td>formData</td>
<td>string</td>
<td>Indicated the format of the rendering: PDF, HTML, REGULATION</td>
</tr>
<tr>
<td>language</td>
<td>formData</td>
<td>string</td>
<td>Optionally, the language of the rendering. The language must be specified in ISO2. For examples: FR, DE, EN ...</td>
</tr>
</tbody>
</table>

#### 2.3.3.2 Response

**2.3.3.2.1 200**

In case of successful response a simple_result containing the rendered notice base64 encoded.

Example:

```json
{
  "result": "JVBERi0xLjQKJa......KNzc0OTYKJSVFT0YK"
}
```

**2.3.3.2.2 400**

In case the received notice parameter is not valid Base64 scheme.

In case the notice parameter or the submission_id parameters are not specified.

In case the rendering format is not valid.

In case the language parameter is not valid.

Example:

```json
{
  "timestamp": "2015-06-17T08:20:30Z",
  "status": 400,
  "error": "Bad Request",
  "exception": "The input is not in valid Base64 scheme",
  "message": "Invalid argument"
}
```

In case for other generic error

Example:

```json
{
  "timestamp": "2016-05-25T18:24:55Z",
  "status": 400,
  "error": "Internal Server Error"
}
```
"error": "Bad Request",
"message": "This error might be caused by a misuse of the API. Please check parameters and API usage according to technical specification",
"path": "/api/production/v1.0/notice/submit",
"error_id": "651dfcdc-efef-47fc-9452-4ad16af9d03"
}

### 2.3.3.2.3 404

In case the related notice could not be found when the `submission_id` parameter is specified.

Example:

```
{
    "timestamp": "2015-06-17T08:21:31Z",
    "status": 404,
    "error": "Not Found",
    "message": "Notice not found"
}
```

In case the related notice could be found but not its content

Example:

```
{
    "timestamp": "2015-06-17T08:21:31Z",
    "status": 404,
    "error": "Not Found",
    "message": "Content not found"
}
```

### 2.3.3.2.4 406

In case the Accept header is provided and set with a value different than ‘application/json’.

Example:

```
{
    "timestamp": "2017-01-10T10:03:30Z",
    "status": 406,
    "error": "Not Acceptable",
    "message": "Not acceptable value for ‘Accept’ header. Only ‘application/json’ format is supported"
}
```

### 2.3.3.2.5 410

In case the related notice could be found but is archived

Example:

```
{
    "timestamp": "2015-06-17T08:21:31Z",
```
"status": 410,
"error": "Gone",
"message": "The requested file has been archived"
}

2.3.3.2.6 422
In case the notice parameter is not valid against the corresponding XSD.
Example:
{
    "timestamp": "2015-06-17T08:22:20Z",
    "status": 422,
    "error": "Unprocessable Entity",
    "exception": "Line:1;Column:835;Error:cvc-complex-type.2.4.a: Invalid content was found starting with element 'NO_DOC_EXTX'. One of '{NO_DOC_EXT}' is expected.",
    "message": "Notice not valid"
}

2.3.3.2.7 503
In case the rendering service is not available.

2.3.4 SEARCH NOTICE

2.3.4.1 URI
GET /notice/search

2.3.4.1.1 Request

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>queryString</td>
<td>string</td>
<td>The status of the notice</td>
</tr>
<tr>
<td>receivedFrom</td>
<td>queryString</td>
<td>string</td>
<td>The start date of the range of notice based on received_at in yyyy/MM/dd date format</td>
</tr>
<tr>
<td>receivedTo</td>
<td>queryString</td>
<td>string</td>
<td>The end date of the range of notice based on received_at in yyyy/MM/dd date format</td>
</tr>
<tr>
<td>pageSize</td>
<td>queryString</td>
<td>int</td>
<td>Indicates the size of the paginated result list</td>
</tr>
<tr>
<td>page</td>
<td>queryString</td>
<td>int</td>
<td>Indicates the page number of the paginated result list</td>
</tr>
<tr>
<td>sort</td>
<td>queryString</td>
<td>string</td>
<td>Indicates the field and the order of the sorting of the paginated result list. For example: submission_id,ASC to sort of the submission_id field in ascending order.</td>
</tr>
</tbody>
</table>

2.3.4.2 Response

2.3.4.2.1 200
In case of successful response a page_result objects is returned.
Example:
{
    "content": [
2.3.4.2.2 400

In case for other generic error

Example:

{  
  "timestamp": "2016-05-25T18:24:55Z",
  "status": 400,
}
"error": "Bad Request",
"message": "This error might be caused by a misuse of the API. Please check parameters and API usage according to technical specification",
"path": "/api/production/v1.0/notice/submit",
"error_id": "651dfcdc-efef-47fc-9452-4ad16af99d03"
}

2.3.4.2.3 406

In case the Accept header is provided and set with a value different than 'application/json'.

Example:
{
  "timestamp": "2017-01-10T10:03:30Z",
  "status": "406",
  "error": "Not Acceptable",
  "message": "Not acceptable value for 'Accept' header. Only 'application/json' format is supported"
}

2.4 Objects

2.4.1 Notice_information

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>submission_id</td>
<td>string</td>
<td>The submission ID generated when the notice is submitted</td>
</tr>
<tr>
<td>received_at</td>
<td>date</td>
<td>Reception date of the notice in eSentool in ISO8601 date-time format</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>status</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVED</td>
<td>Acknowledgement of receipt. The notice is being checked in the reception system.</td>
<td>X</td>
</tr>
<tr>
<td>RECEPTION_ERROR</td>
<td>In case the notice could not be validated, and the publication process could not be started</td>
<td>X</td>
</tr>
<tr>
<td>QUALIFICATION_ERROR</td>
<td>In case the notice could not be validated for</td>
<td>X</td>
</tr>
<tr>
<td>reason_code</td>
<td>Status</td>
<td>Reason code</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>validation</td>
<td>VALIDATION_ACCEPTED</td>
<td>BV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XMLV</td>
</tr>
<tr>
<td></td>
<td>QUALITY_ACCEPTED</td>
<td>CP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPV</td>
</tr>
<tr>
<td></td>
<td>QUALITY_SKIPPED</td>
<td>DU</td>
</tr>
<tr>
<td></td>
<td>IN_PROGRESS</td>
<td>HR</td>
</tr>
<tr>
<td></td>
<td>PUBLISHED</td>
<td>NOT_PUBLISHED</td>
</tr>
<tr>
<td></td>
<td>NOT_PUBLISHED</td>
<td>WAITING_FOR_INFORMATION</td>
</tr>
<tr>
<td></td>
<td>WAITING_FOR_INFORMATION</td>
<td></td>
</tr>
<tr>
<td>status_updated_at</td>
<td>date</td>
<td>Date of the update of the current status in ISO8601 date-time format</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>no_doc_ext</td>
<td>string</td>
<td>Unique identifier of the notice in the eSender system. A string matching the pattern: YYYY-nnnnnn. Examples: 2015-123456, 2016-000001, or 2017-999999</td>
</tr>
<tr>
<td>form</td>
<td>string</td>
<td>The type of form of the notice: F01, F02, ...</td>
</tr>
<tr>
<td>languages</td>
<td>string array</td>
<td>The list of languages of the notice</td>
</tr>
<tr>
<td>publication_info</td>
<td>publicatio n_info</td>
<td>The information of publication of the notice</td>
</tr>
<tr>
<td>technical_validation_report</td>
<td>validation _report</td>
<td>The validation report of the notice generated during the technical checking list</td>
</tr>
<tr>
<td>validation_rules_report</td>
<td>validation _report</td>
<td>The validation report of the notice generated during the validation rules checks</td>
</tr>
<tr>
<td>quality_control_report</td>
<td>validation _report</td>
<td>The validation report of the notice generated during the quality control checks</td>
</tr>
<tr>
<td>ref_submission_id</td>
<td>string</td>
<td>The Submission Id of the referenced notice, if any.</td>
</tr>
<tr>
<td>ref_no_doc_ojs</td>
<td>string</td>
<td>The publication number of the referenced notice, if published.</td>
</tr>
</tbody>
</table>
### 2.4.2 Publication Info

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ojs_number</td>
<td>string</td>
<td>The OJS number of the notice as published on TED website. Examples: 001, 128, 999</td>
</tr>
<tr>
<td>publication_date</td>
<td>date</td>
<td>The publication date of the notice on TED website.</td>
</tr>
<tr>
<td>ted_links</td>
<td>ted_links array</td>
<td>The list of TED website links for each language of the notice</td>
</tr>
</tbody>
</table>

### 2.4.3 TED Links

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>The language code of the TED website link</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The TED website link</td>
</tr>
</tbody>
</table>

### 2.4.4 Validation Report

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>The type of validation report, can be one of TECH, VALIDATION_RULES, QUALITY_CONTROL</td>
</tr>
<tr>
<td>items</td>
<td>validation_report_items array</td>
<td>A list of validation_report_items</td>
</tr>
</tbody>
</table>

### 2.4.5 Validation Report Items

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name of the validation rule</td>
</tr>
<tr>
<td>valid</td>
<td>boolean</td>
<td>Indicates if the validation rule successfully passed</td>
</tr>
<tr>
<td>severity</td>
<td>String</td>
<td>In case the validation rule failed, indicated the severity of the rule: INFO, WARNING, ERROR, CRITICAL</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>A message to briefly describe the error</td>
</tr>
<tr>
<td>details</td>
<td>string</td>
<td>A complete error message</td>
</tr>
</tbody>
</table>

### 2.4.6 Simple Result

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>string</td>
<td>The result of the operation, this can be a message or a base64 payload such as a PDF.</td>
</tr>
</tbody>
</table>
## 2.4.7 PAGE_RESULT

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>array of notice_information</td>
<td>Contains a page of search results</td>
</tr>
<tr>
<td>total_elements</td>
<td>int</td>
<td>The total number of elements in this search results</td>
</tr>
<tr>
<td>last</td>
<td>boolean</td>
<td>Whether this page is the last page</td>
</tr>
<tr>
<td>total_pages</td>
<td>int</td>
<td>Total number of pages</td>
</tr>
<tr>
<td>size</td>
<td>int</td>
<td>The page size</td>
</tr>
<tr>
<td>number</td>
<td>int</td>
<td>The number of the current page, starting from 0</td>
</tr>
<tr>
<td>sort</td>
<td>string</td>
<td>The actual sorting of the elements</td>
</tr>
<tr>
<td>number_of_elements</td>
<td>int</td>
<td>Number of elements in this page</td>
</tr>
<tr>
<td>first</td>
<td>boolean</td>
<td>Whether this page is the first page</td>
</tr>
</tbody>
</table>
2.5 Notice life cycle

Below is a diagram showing the life cycle of a Notice.

*eSentool – Notice life cycle*

![Notice life cycle diagram](image)

*Figure 1: Notice life cycle*

2.6 JAVA REST Client

A JAVA client could be used in order to ease the request to the eSentool REST API. This library has external dependencies with org.apache.httpcomponents:httpclient:4.4.1 and com.fasterxml.jackson.core:jackson-databind:2.5.3.

2.6.1 Create an eSentool REST Client instance

```java
new EsentoolRestClientBuilder("https://hostname/qualification/rest/v1.0/", "TED123", "password").build();
```

2.6.2 Call one of the REST operation

```java
Page<NoticeInformation> result = client.searchNotices(new SearchNoticeParams().withPage(0).withPageSize(15).withSort("submission_id").withSortDirection(SortDirection.ASC));
```