made with europeana

Enriching Europeana with user transcriptions and annotations (EnrichEuropeana)

Milestone 7: Final version of automatic enrichment and quality control services deployed

1. INFORMATION ON THE ACTION	
Grant Agreement Nº	INEA/CEF/ICT/A2017/1568419
Action Title (Art. 1 of G.A.)	Enriching Europeana with user transcriptions and annotations (EnrichEuropeana)
Action number (Art. 1 of the G.A.)	2017-EU-IA-0142

Editorial Information	
Revision	1.0
Date of submission	
Author(s)	Srdjan Stevanetic,Mihai Bartha, Sergiu Gordea
Dissemination Level	public

Revision History

Revision No.	Date	Author	Organization	Description
0.1	03.12.2019	Srdjan Stevanetic	AIT	first contributions to all sections of the document
0.2	23.12.2019	Mihai Bartha	AIT	document review
1.0	17.01.2020	Sergiu Gordea	AIT	final editing

Statement of originality

Co-financed by the Connecting Europe Facility of the European Union

This report contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The contents of this publication are the sole responsibility of the authors and do not necessarily reflect the opinion of the European Union.

Table of contents

Introduction **3**
Enrichment Workflow 4
Service Description **5**
Translation services 5
Translation POST request for stories 5
Retrieve translations of stories 7
Request for performing machine translation of Transcribathon items 8
Retrieve translations of item textual properties 10
Enrichment Services 11
Retrieve enrichments request for stories 12
Compute enrichments request for items 13
Retrieve enrichments request for items 15
Retrieve individual enrichment for stories and items 16
Named Entity Recognition for stories 18
Retrieve identified Named Entities for stories 20
Named Entity Recognition for items 22
Enrichment Named Entity Recognition GET request for items 25
Entity Preview Services 26
Search known entities 26
Retrieve entity preview 28
Administration services 30
Administration POST service for importing stories and items from JSON 30
Administration POST service for importing stories over the request body field 31
Administration POST service for uploading items over the request body field 31
Administration POST service for uploading the translations of items or stories 33
Integration of Enrichment Services in Transcribathon Tool **36**
Conclusions and further Development **37**
Annex 1: Tools used for the evaluation of Named Entity Recognition, classification and linking **37**

Co-financed by the Connecting Europe Facility of the European Union

Introduction

This document presents the technical documentation for the automatic enrichment services developed within the scope of **Activity 2: Semantic Enrichment and Quality Control Services.**

Manual transcription of text materials, as enabled by the Transcribathon platform, generates high-accuracy resources, supports a better understanding of materials and allows further reuse. The goal of the newly developed platform is to enhance the accessibility and understandability of material related to overarching themes which hold relevance to important events that contributed to the development of modern Europe, namely World War I, Migration or Revolutions of 1989. New technologies for the automated analysis of transcribed text are developed to achieve this goal. Machine-translation services are integrated in order to make the content accessible and searchable for users in their preferred languages. Natural language processing technology is employed to automatically extract named entities and to match them against semantic web repositories like Europeana Entity Collection, DbPedia and/or Wikidata. Translations are also used as an enabler for integrating advanced solutions for extraction of contextual information from historical documents.

The work presented in this document includes the progress achieved within the following Tasks of the Action (with respect to the changes against the previous milestone):

Task 2.1. Controlled vocabularies for transcription and metadata enrichments
The semantic enrichment of textual resources and metadata is based on usage of linked data resources, which will be accessed through a unique endpoint represented by Europeana Entity API. For this purpose, relevant named entities from open linked data repositories (e.g. Wikidata, DbPedia, Geonames, etc.) will be integrated within the Europeana Entity Collection (EC). The EC is the underlying knowledge graph developed by Europeana by combining statements from various linked data repositories. It has the goal to centralise information about the contextual entities related to the cultural heritage objects. New named entities identified by Task 2.3 will be prepared and proposed for ingestion in EC which can then be made available through the Europeana Entity API. This way, the new entities will be available to be used for semantic enrichment of cultural heritage objects, displayed in Europeana as Entity Pages, and used for auto-suggestion and search, both in Europeana Collections and EnrichEuropeana platform.

Task 2.2. Pilot on automatic translation of transcribed text
The textual resources generated through the transcription campaigns will be translated into all official European languages by using the translation services provided by the eTranslation DSI and Google Translate. The automatic translations will be stored in a database and in a multilingual search index (i.e. using Apache Solr), and made available to the application developers through the new developed Translation API. This way, the API will provide support for the retrieval of cultural heritage objects (i.e. transcribed manuscripts) by using the search terms in any European language. The translations of the transcribed text will be used as input for the Natural Language Processing services developed in Task 2.3.

Task 2.3. Automatic analysis and Named Entity Recognition
Natural language processing algorithms will be used to analyse the German and English transcriptions and translations (i.e. Natural Language Processing models are language specific). By employing Named Entity Recognizers such as Stanford NER (See https://nlp.stanford.edu/software/CRF-NER.html), this task will discover named entities within the transcribed text or in their translations. In the following step, the identified entities will be searched within the Europeana Entity Collection using the search and auto-suggestion functionality of the Entity API. The new entities that are not yet available in the Entity Collection, will be searched within the Linked Data repositories, such as Wikidata, DBPedia, Geonames, and proposed for ingestion within the scope of Task 2.1. All resolvable named entities will be used for semantic enrichment of cultural heritage objects by using their Europeana URIs or their Linked Data URIs.

Task 2.4. Quality control and validation
The accuracy of semantic enrichments and translations will be enhanced through manual verification and crowdsourcing. Currently, the moderation functionality in the Europeana Annotations API provides support for disapproving and automatic disabling of semantic enrichments (i.e. automatic or user generated annotations). This functionality will be enhanced to support the approval of automatic enrichments by regular end users or application moderators. Starting from the automatic translations of transcribed text, end users will have the possibility to provide improved text translations and to validate them by using the same approval process. Only validated semantic enrichments and translations (i.e. those that are confirmed of being correct) will be visible for the public information consumers.

Co-financed by the Connecting Europe
Facility of the European Union

Enrichment Workflow

Figure 1. shows the enrichment workflow diagram. The workflow steps have not been changed from the previous deliverable but they have been improved with regard to both functional and non-functional requirements. Under changes in functional requirements, we mean the changes in the functionality of the services (e.g. adding the data retrieval from the Transcribathon platform in case of missing data or providing a local cache of data from semantic repositories like Wikidata). Under changes in non-functional requirements, we mean the changes in the quality attributes of the services (e.g. performance or modularity improvements).

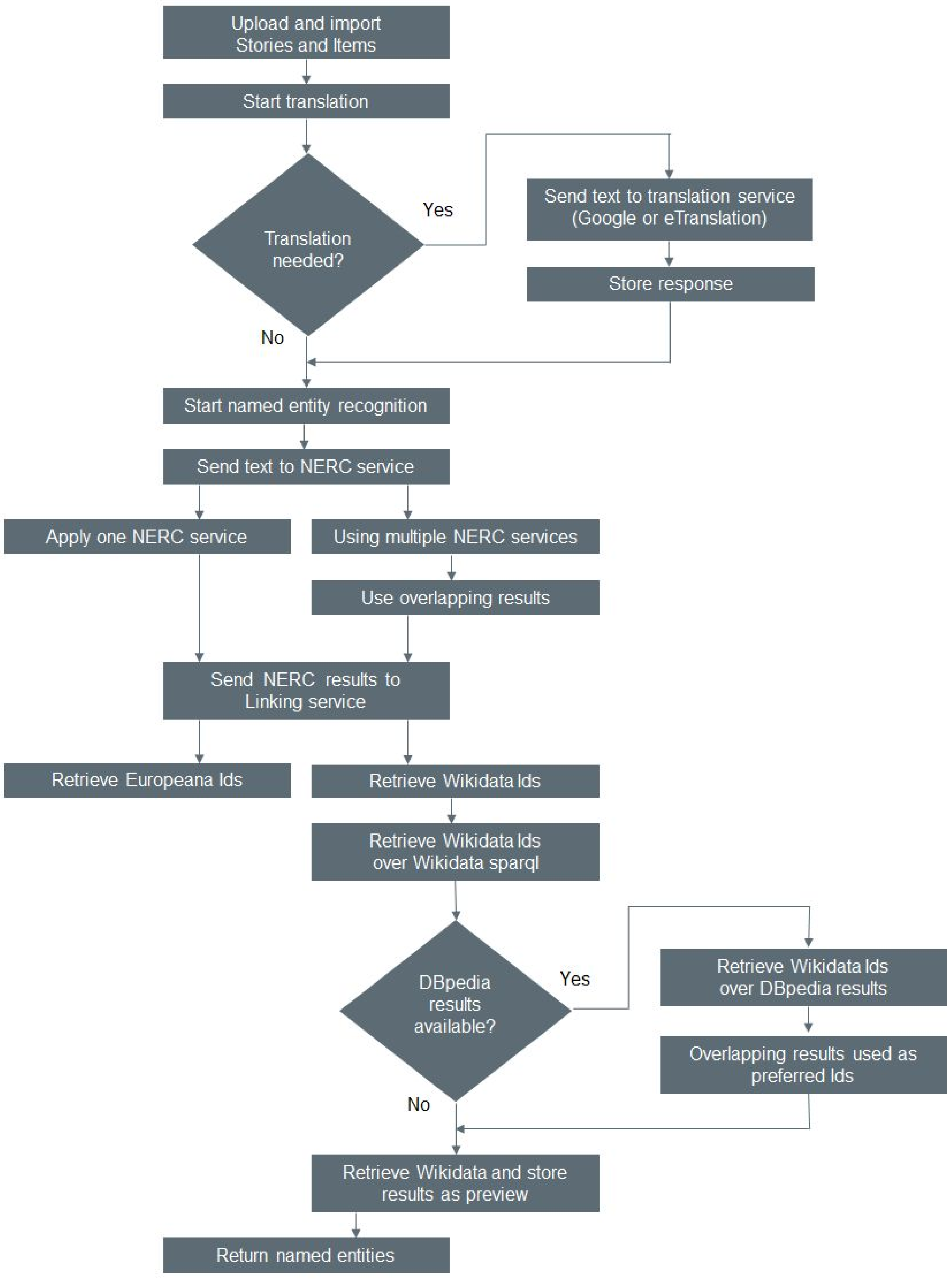

Upload and import Stories and Items
Start translation
Translation needed?
Yes
No
Send text to translation service (Google or eTranslation)
Store response
Start named entity recognition
Send text to NERC service
Apply one NERC service
Using multiple NERC services
Use overlapping results
Send NERC results to Linking service
Retrieve Europeana Ids
Retrieve Wikidata Ids
Retrieve Wikidata Ids over Wikidata sparql
DBpedia results available?
Yes
No
Retrieve Wikidata Ids over DBpedia results
Overlapping results used as preferred Ids
Retrieve Wikidata and store results as preview
Return named entities

Figure 1: Enrichment Workflow (reused from the previous deliverable)

The enrichment workflow shown in Figure 1. includes the following data processing steps (please note that some text in the steps descriptions below may overlap with the text in the previous milestone since the changes are built upon the already existing implementation and the holistic description of all steps is necessary for the sake of completeness):

- **Upload stories and items**: The Transcribathon stories and items can be manually imported in the database of Enrichment services. The import APIs remain the same as described in the previous milestone version (see below Section Admin

Console) except that the functionality is adapted so that existing stories and items can be modified by importing stories and items with the same storyId or itemId where the modifications are provided for the other story or item parameters. Additionally, stories and items are imported to the Solr server using the Solr import handlers. The transcription of a story and/or item is processed for eliminating the HTML markup used in the Transcribathon platform.

- **Start translation:** The descriptions and transcriptions of stories and items can be translated to English in order to make them available for the Named Entity Recognition (NER) processing, to enhance the accessibility and to improve discoverability through the search functionality. NER processing can also be pursued on the original text only if stories and items are in English or German language. Google Translate and eTranslation services are available to be used for translating documents, and the translations are stored into the database and Solr index as well. Currently, we have translated all items and imported the translations into the database and the Solr index. With regard to the previous version of the APIs, the translations can be separately done for stories and for items and there is a new parameter called *text* where the user can specify the text for the given story or item *property* (description or transcription) by himself (see below for the APIs details). Also, if the given story or item is not present in the database, the same is retrieved from the Transcribathon platform.
- **Start NER**: The invocation of named entity recognition and classification (NERC) processing can be run separately for the individual services available (i.e. apply one NERC service), or multiple services can be run with one command (i.e. using multiple NERC services). In the second case, the results provided by individual services will be cross-validated and only the non-ambiguous results will be used in the following processing steps. The output of this processing step includes the labels of the recognized named entities and their categorization. While more classes (i.e., categories) of entities are identified by the employed tools, only Persons and Locations are considered to be relevant for the enrichment of historical documents. Changes with regard to the previous version of the services encompass fetching stories and items from the Transcribathon platform if they are not present in the database, and translating them on the fly if they are still not translated. Furthermore, new APIs for separate processing of stories and items have been developed (see below for details).
- **Send NERC results to the linking service**: The linking service resolves entities against items in semantic repositories by using their labels. While DBpedia Spotlight resolves itself entities against language-specific DBpedia versions (e.g. English or German DBpedia), the service fetches the data from the repository to retrieve the identifier (URI) of the Wikidata resource, that is considered to be the primary source of information. In contrast, the NLP based pipeline is using the recognized entity name, its category and the languages of the text to search for entries in the Wikidata repository. The *suggest* method of the Europeana Entity API is used to retrieve the URIs for the entities of the Europeana Entity Collection.
- **Retrieve Wikidata resource descriptions:** The description of linked resources is retrieved from Wikidata in order to generate entity previews. The format used for representing the previews is based on the information included in the *suggest* method of Europeana Entity API. These previews may be used by end users to verify the correctness of enrichments that are automatically generated. New APIs for searching the named entities found in the process of named entity recognition as well as for searching the Wikidata places based on Solr queries have been provided (see below for details). A unique set of around 180000 Wikidata places from the Europeana platform is imported to the Solr index, to serve as a local cache of places for linking with the found named entities during the NER process.
- **Generate Semantic Enrichments:** The final step of the processing workflow consists of the serialization of automatic enrichments using Web Annotation Data Model, following the representation supported by the Europeana Annotations API. Currently, only enrichments with higher confidence (acquired through the cross-checking results of the two processing pipelines) are exposed.

Each step from the enrichment process can be individually performed by using the REST API (both GET and POST) described in Section Service Description.

Service Description

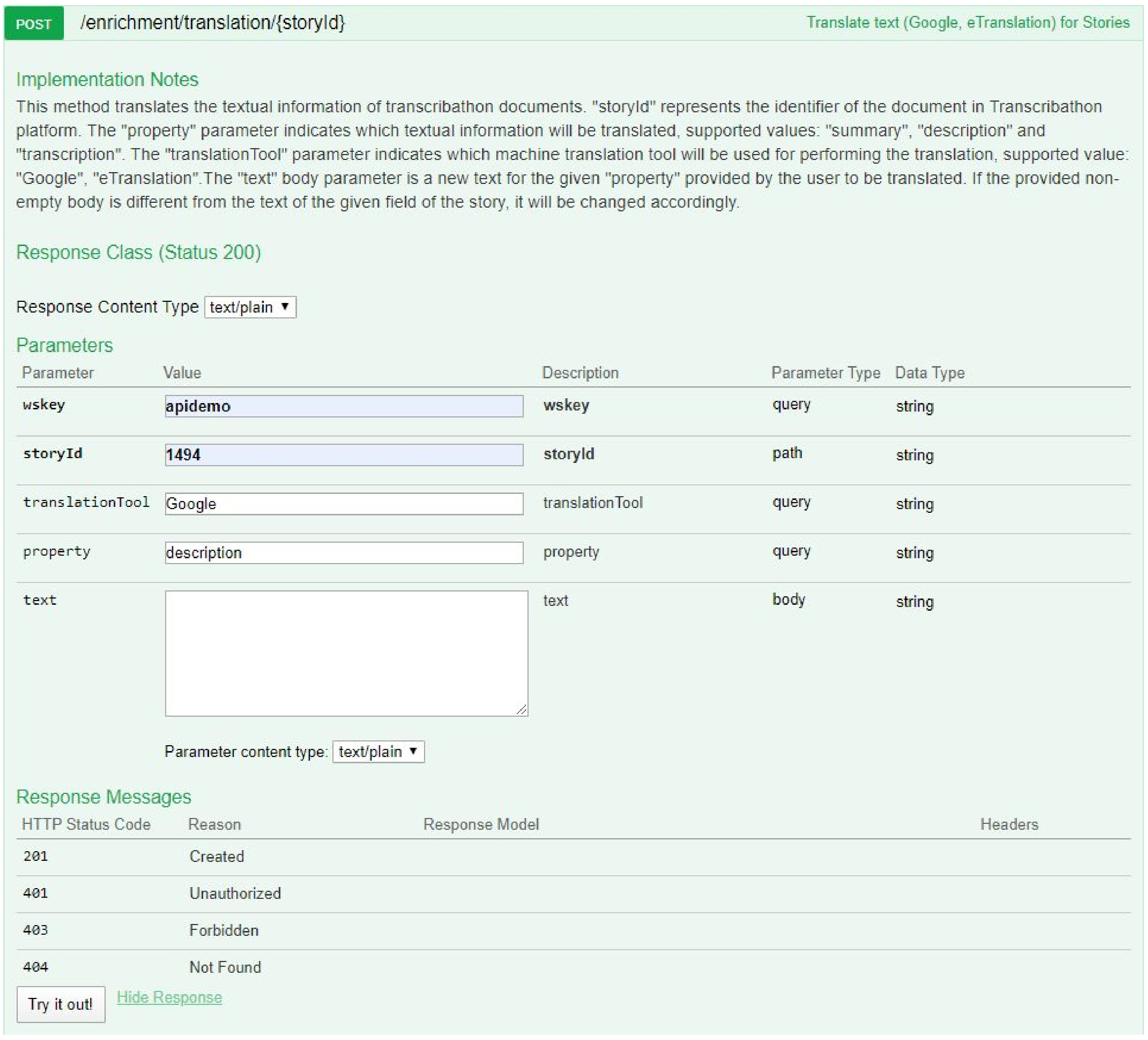
The specifications of all services follow the conventions used to develop the Europeana Core Services APIs. It means they are all exposed through REST interfaces and integrated and run within a Swagger console which enables developers to implement and test individual services using the corresponding HTTP requests. The specifications of the semantic enrichment services were extended with additional requirements with regard to the integration in the Transcribathon platform. In the following, the functionality implemented by the individual API methods is presented together with sample invocations through the provided Swagger UI.

Translation services

Translation POST request for stories

The machine translation of Transcribathon stories is performed through the submission of a POST request as indicated in Figure 2. The story to be translated is selected using the “storyId” parameter, where the textual property to be translated is indicated though the “property” parameter (one of description, summary or transcription). The “translationTool” parameter indicates which service is invoked to perform the actual machine translation (either Google or eTranslation). Optionally, the text to be translated can be directly submitted through the request body. If the body is left empty the already stored text is used. Figure 2 shows the Swagger UI of the service with the description of all parameters and their types.

In the process of translation either eTranslation or Google translation APIs are called programmatically. The translated text is then saved into the database. In case that the given translation already exists in the database, it is retrieved and returned in the response. A hash value of the translation is also saved in order to enable easy comparisons among the translations texts when some translation is updated with a new text. In case that some parameters are not valid or the given storyId does not exist in the Transcribathon Platform an error response is returned together with an appropriate explanation.



POST /enrichment/translation/{storyId} Translate text (Google, eTranslation) for Stories

Implementation Notes

This method translates the textual information of transcribathon documents. "storyId" represents the identifier of the document in Transcribathon platform. The "property" parameter indicates which textual information will be translated, supported values: "summary", "description" and "transcription". The "translationTool" parameter indicates which machine translation tool will be used for performing the translation, supported value: "Google", "eTranslation".The "text" body parameter is a new text for the given "property" provided by the user to be translated. If the provided non-empty body is different from the text of the given field of the story, it will be changed accordingly.

Response Class (Status 200)

Response Content Type text/plain

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
translationTool	Google	translationTool	query	string
property	description	property	query	string
text		text	body	string

Parameter content type: text/plain

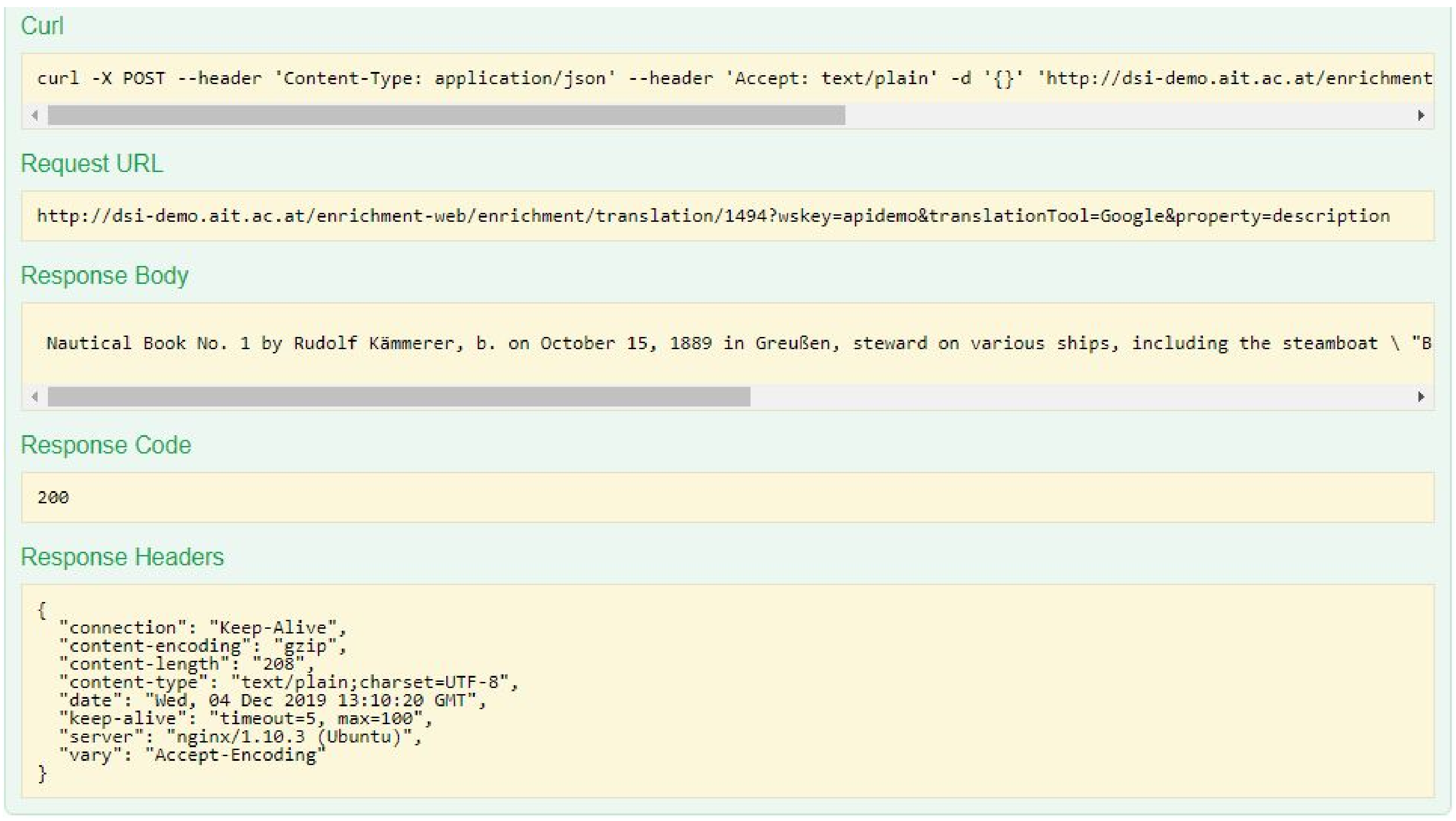
Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Figure 2: Request for machine translation of stories

Figure 3 shows the swagger UI with the actual request submitted by the console and the response retrieved when the parameters shown in Figure 2 are used. In the Response Body part the translated text is returned.

Co-financed by the Connecting Europe Facility of the European Union



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' -d '{}' 'http://dsi-demo.ait.ac.at/enrichment
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494?wskey=apidemo&translationTool=Google&property=description
```

Response Body

```
 Nautical Book No. 1 by Rudolf Kämmerer, b. on October 15, 1889 in Greußen, steward on various ships, including the steamboat \ "B
```

Response Code

```
200
```

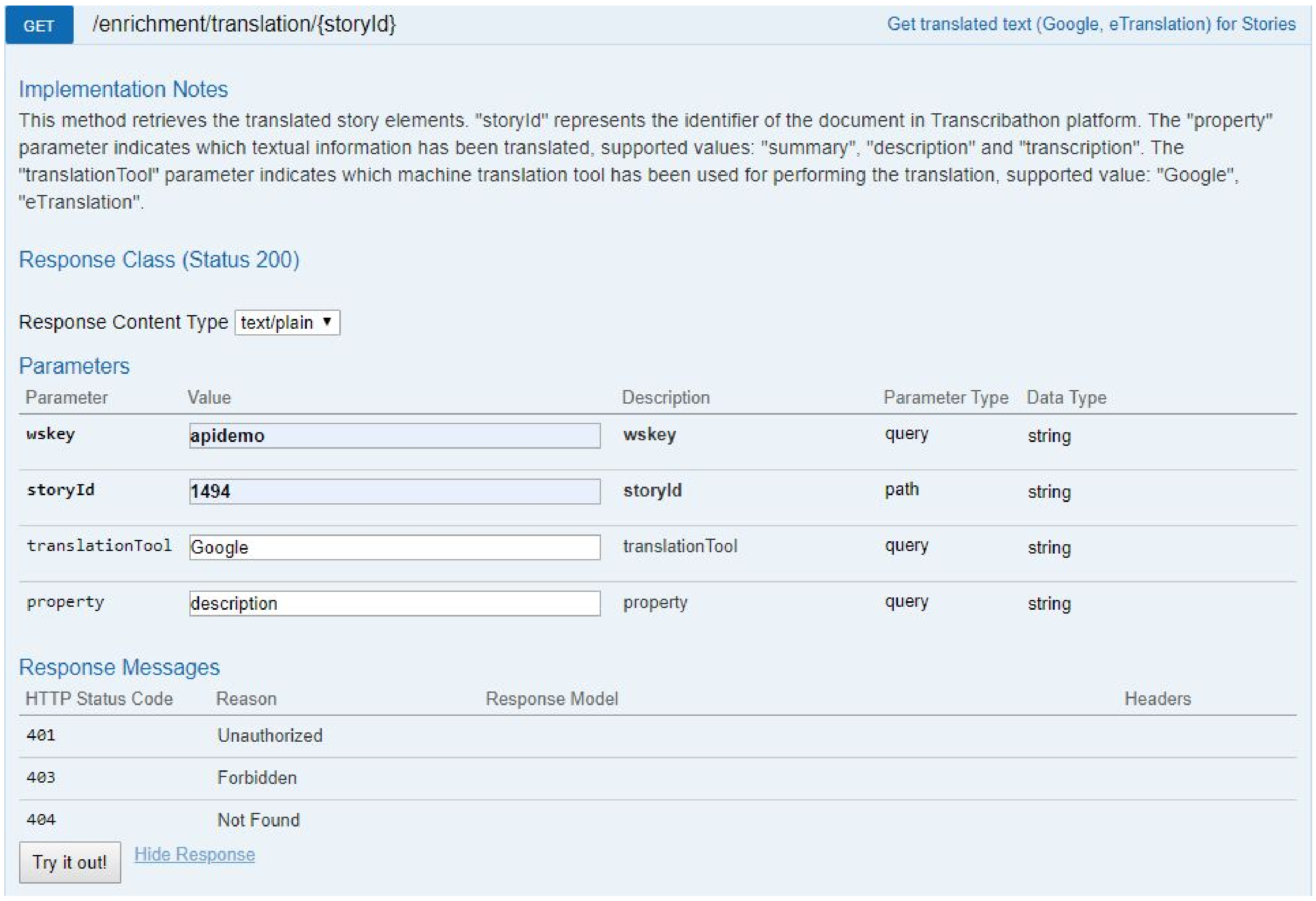
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-encoding": "gzip",
  "content-length": "208",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Wed, 04 Dec 2019 13:10:20 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)",
  "vary": "Accept-Encoding"
}
```

Figure 3: Response for the machine translation of stories

Retrieve translations of stories

Figure 4 shows the request for translation of stories within the swagger UI interface. The retrieve translation request uses the same parameters as the request for performing machine translation. This service retrieves the translation of the given story from the database provided that the translation is previously performed with the given POST method. In case that the translation has not been performed yet, an HTTP exception Precondition Required is thrown with the information that the translation of the story first need to be done with the given POST request.



GET /enrichment/translation/{storyId} Get translated text (Google, eTranslation) for Stories

Implementation Notes

This method retrieves the translated story elements. "storyId" represents the identifier of the document in Transcribathon platform. The "property" parameter indicates which textual information has been translated, supported values: "summary", "description" and "transcription". The "translationTool" parameter indicates which machine translation tool has been used for performing the translation, supported value: "Google", "eTranslation".

Response Class (Status 200)

Response Content Type text/plain

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
translationTool	Google	translationTool	query	string
property	description	property	query	string

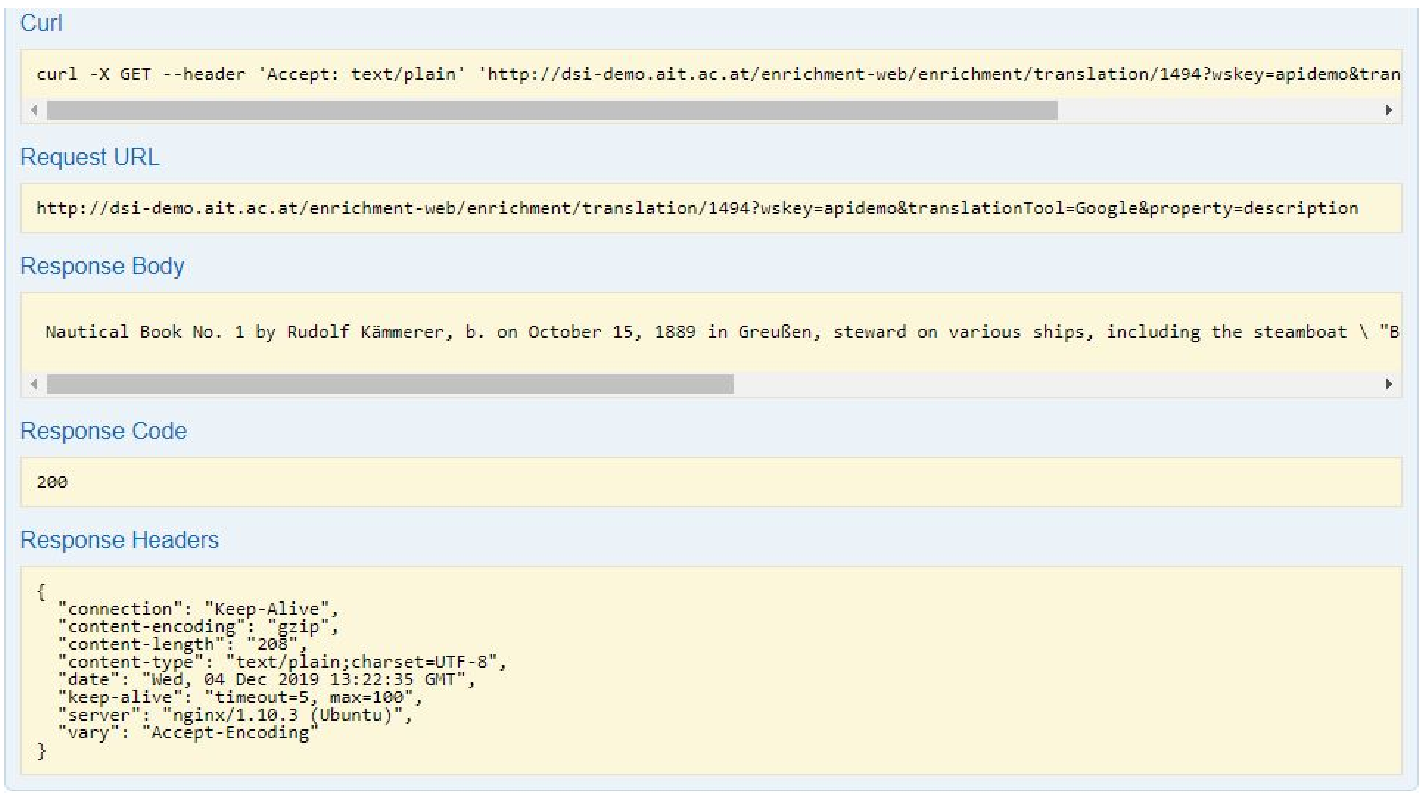
Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Figure 4: Request to retrieve previously translated stories

Figure 5 shows the response of the retrieve translation request with the parameters indicated in Figure 4.The translation is available in the Response Body. In case that some parameters are invalid the same errors are shown as in the case of the corresponding POST request.

Co-financed by the Connecting Europe
Facility of the European Union



Curl

```
curl -X GET --header 'Accept: text/plain' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494?wskey=apidemo&tran
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494?wskey=apidemo&translationTool=Google&property=description
```

Response Body

```
 Nautical Book No. 1 by Rudolf Kämmerer, b. on October 15, 1889 in Greußen, steward on various ships, including the steamboat \ "B
```

Response Code

```
200
```

Response Headers

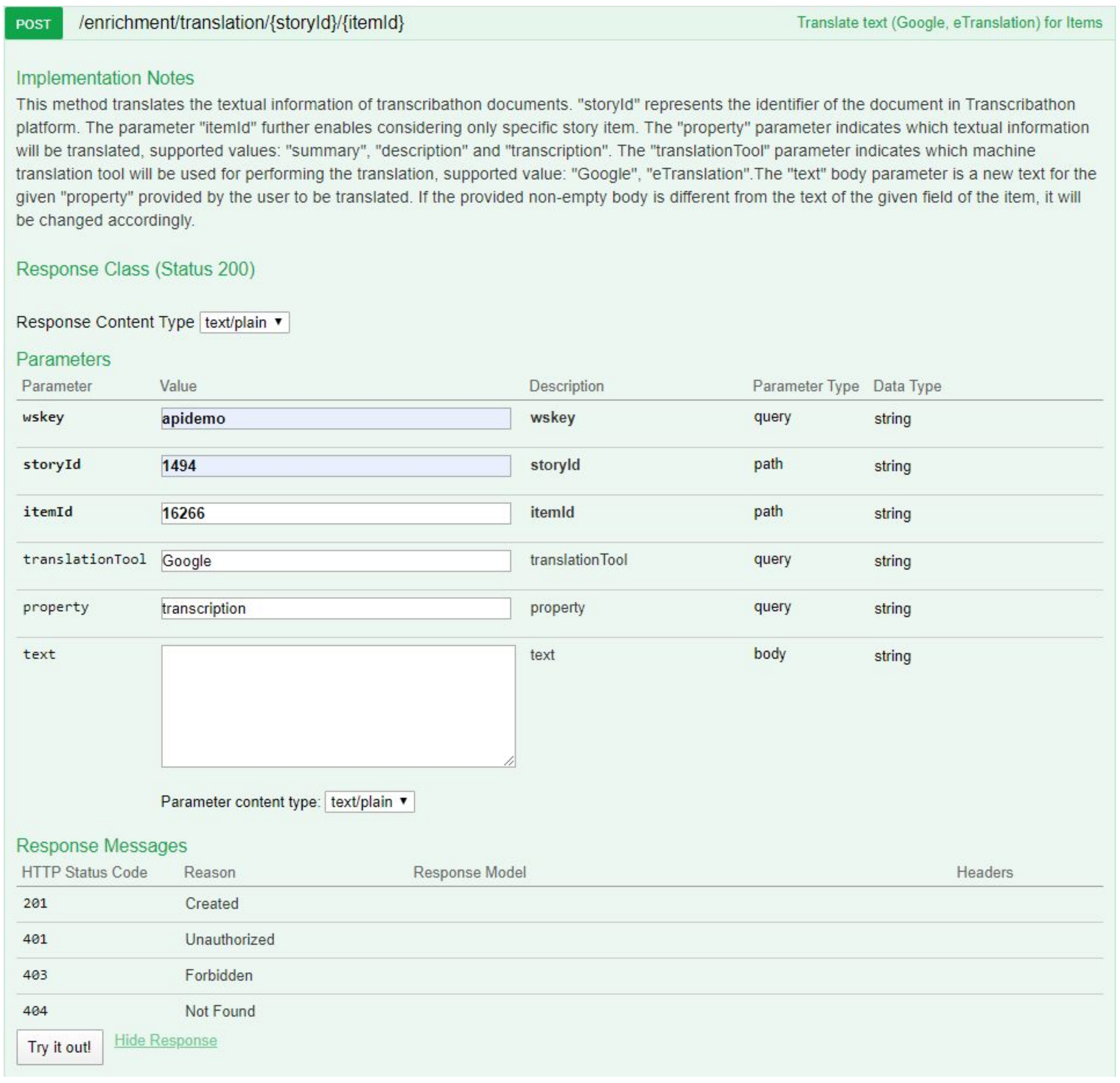
```
{
  "connection": "Keep-Alive",
  "content-encoding": "gzip",
  "content-length": "208",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Wed, 04 Dec 2019 13:22:35 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)",
  "vary": "Accept-Encoding"
}
```

Figure 5: Response for the retrieve translation of stories request

Request for performing machine translation of Transcribathon items

Typically, a Transcribaton story represents a document, while and item represents an individual page of the document. The requests for translation of Transcribathon items are designed to be consistent with the ones for the translation of stories, except that only the text related to a given is translated. Using the “itemId” parameter, the user indicates the item for which the textual information is translated. Figure 6 showcases a request for translating a transcription of a given item. When the request body is left empty the value of the transcription stored in the database is used for the translation. Figure 7 shows the corresponding request and response data for the input data from Figure 6.

Co-financed by the Connecting Europe
Facility of the European Union



POST /enrichment/translation/{storyId}/{itemId} — Translate text (Google, eTranslation) for Items

Implementation Notes

This method translates the textual information of transcribathon documents. "storyId" represents the identifier of the document in Transcribathon platform. The parameter "itemId" further enables considering only specific story item. The "property" parameter indicates which textual information will be translated, supported values: "summary", "description" and "transcription". The "translationTool" parameter indicates which machine translation tool will be used for performing the translation, supported value: "Google", "eTranslation".The "text" body parameter is a new text for the given "property" provided by the user to be translated. If the provided non-empty body is different from the text of the given field of the item, it will be changed accordingly.

Response Class (Status 200)

Response Content Type: text/plain

Parameters

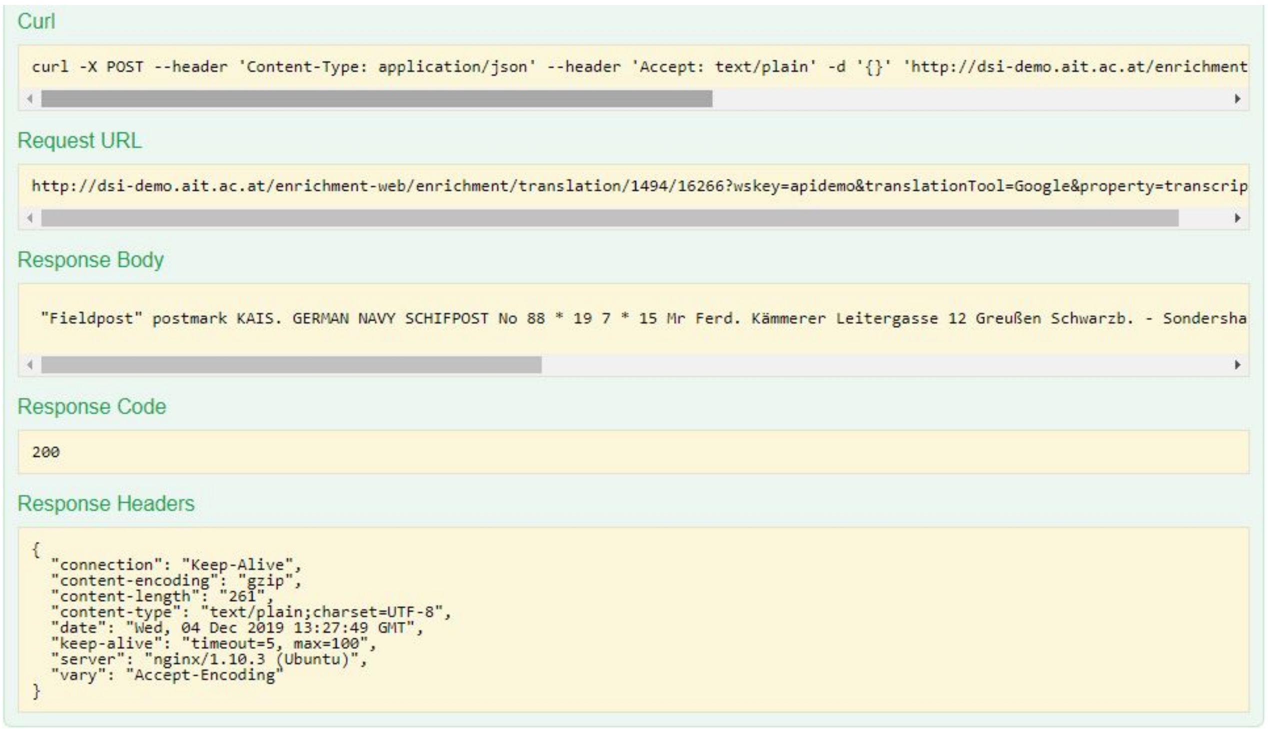
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
itemId	16266	itemId	path	string
translationTool	Google	translationTool	query	string
property	transcription	property	query	string
text		text	body	string

Parameter content type: text/plain

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Figure 6: Request for the machine translation of items



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' -d '{}' 'http://dsi-demo.ait.ac.at/enrichment
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494/16266?wskey=apidemo&translationTool=Google&property=transcrip
```

Response Body

```
"Fieldpost" postmark KAIS. GERMAN NAVY SCHIFPOST No 88 * 19 7 * 15 Mr Ferd. Kämmerer Leitergasse 12 Greußen Schwarzb. - Sondersha
```

Response Code

```
200
```

Response Headers

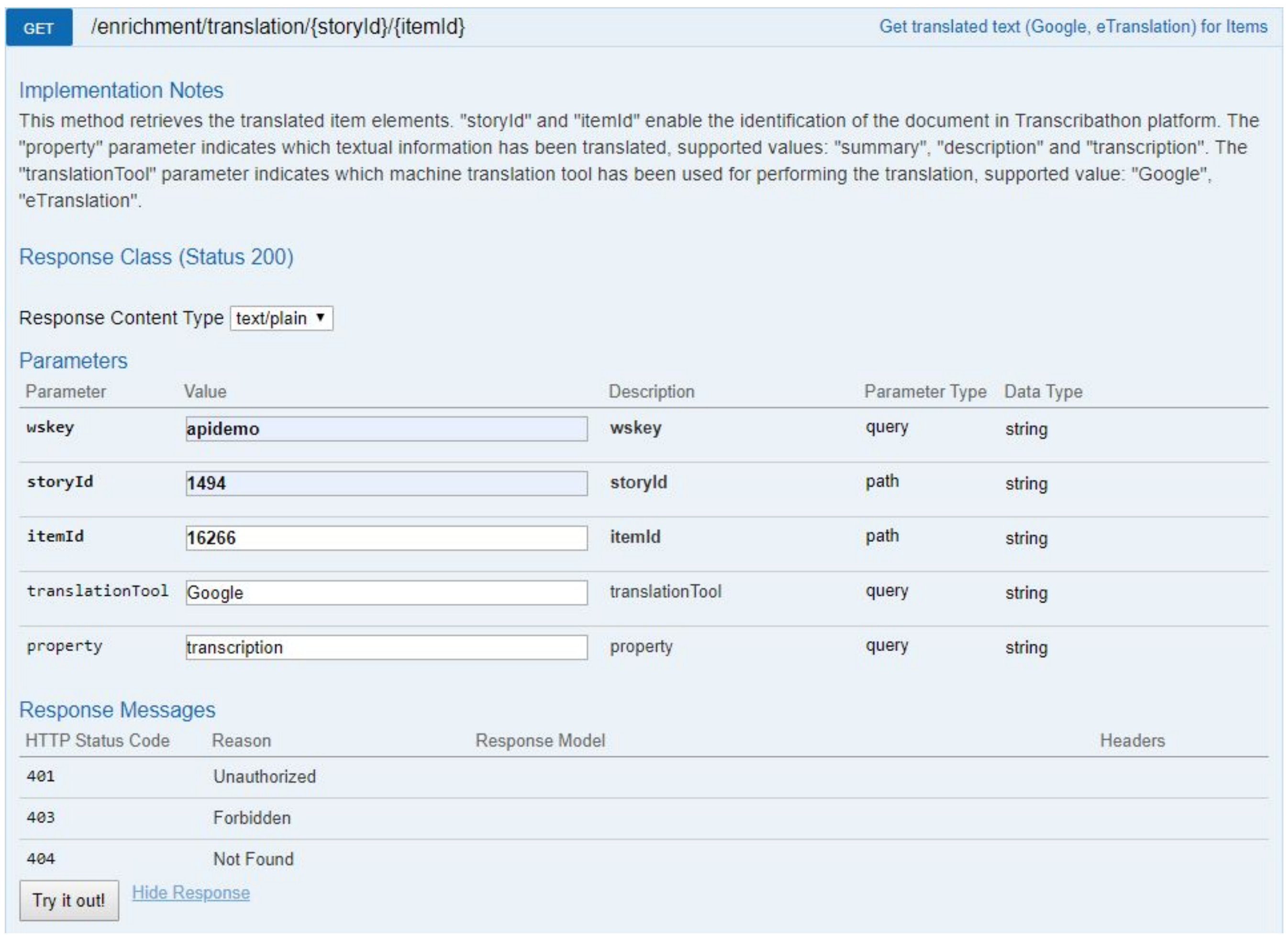
```
{
  "connection": "Keep-Alive",
  "content-encoding": "gzip",
  "content-length": "261",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Wed, 04 Dec 2019 13:27:49 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)",
  "vary": "Accept-Encoding"
}
```

Figure 7: Response for the machine translation of items request

Retrieve translations of item textual properties

The translation GET request for items behaves in the same way as the corresponding GET request for stories except that it retrieves the item translations from the database based on the given “itemId” parameter. Figure 8 shows a swagger UI interface for retrieving items. Figure 9 shows the corresponding request and response data for the input parameters from Figure 8.

Co-financed by the Connecting Europe Facility of the European Union



GET /enrichment/translation/{storyId}/{itemId}   Get translated text (Google, eTranslation) for Items

Implementation Notes

This method retrieves the translated item elements. "storyId" and "itemId" enable the identification of the document in Transcribathon platform. The "property" parameter indicates which textual information has been translated, supported values: "summary", "description" and "transcription". The "translationTool" parameter indicates which machine translation tool has been used for performing the translation, supported value: "Google", "eTranslation".

Response Class (Status 200)

Response Content Type text/plain

Parameters

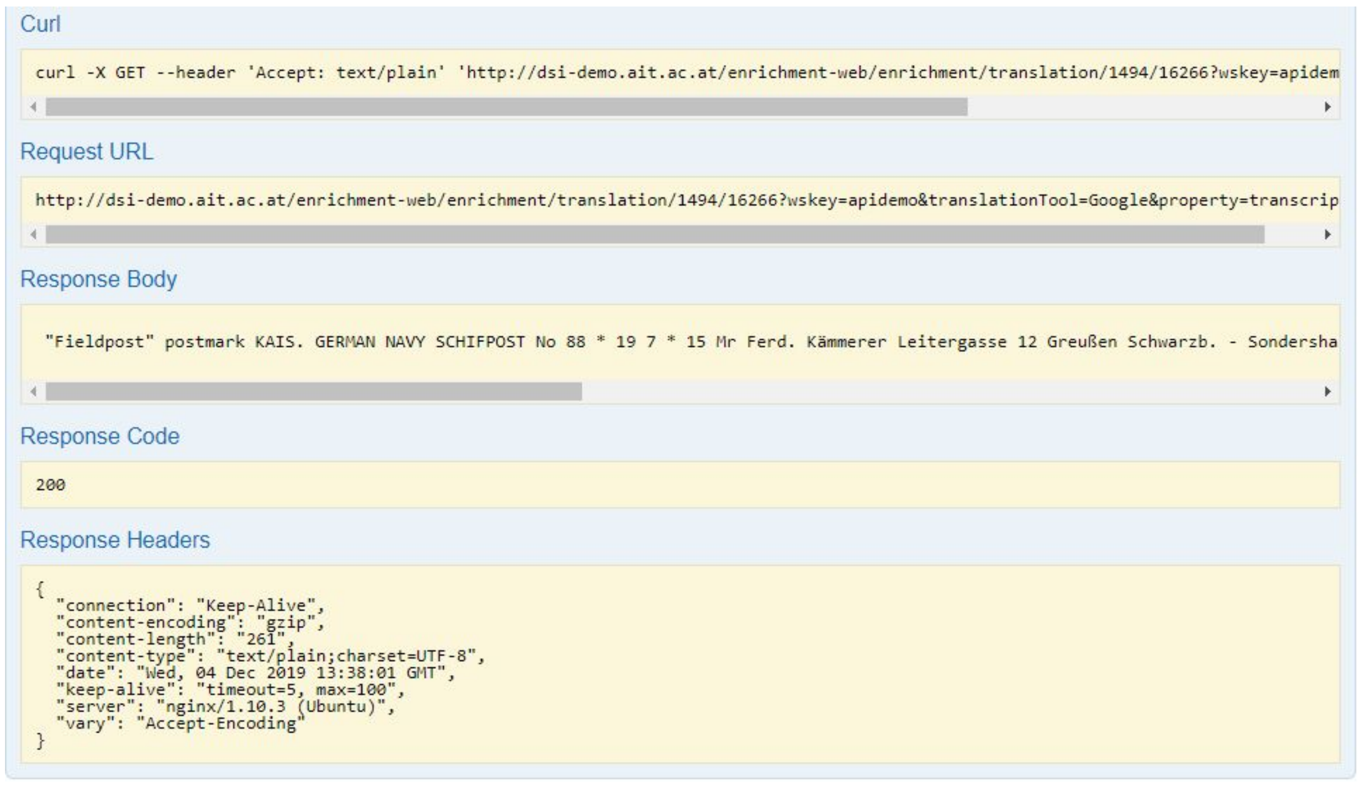
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
itemId	16266	itemId	path	string
translationTool	Google	translationTool	query	string
property	transcription	property	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Figure 8: Retrieve translations for items request



Curl

```
curl -X GET --header 'Accept: text/plain' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494/16266?wskey=apidem
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/translation/1494/16266?wskey=apidemo&translationTool=Google&property=transcrip
```

Response Body

```
 "Fieldpost" postmark KAIS. GERMAN NAVY SCHIFPOST No 88 * 19 7 * 15 Mr Ferd. Kämmerer Leitergasse 12 Greußen Schwarzb. - Sondersha
```

Response Code

```
200
```

Response Headers

```
{
  "connection": "Keep-Alive",
  "content-encoding": "gzip",
  "content-length": "261",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Wed, 04 Dec 2019 13:38:01 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)",
  "vary": "Accept-Encoding"
}
```

Figure 9: Retrieve translations for items response

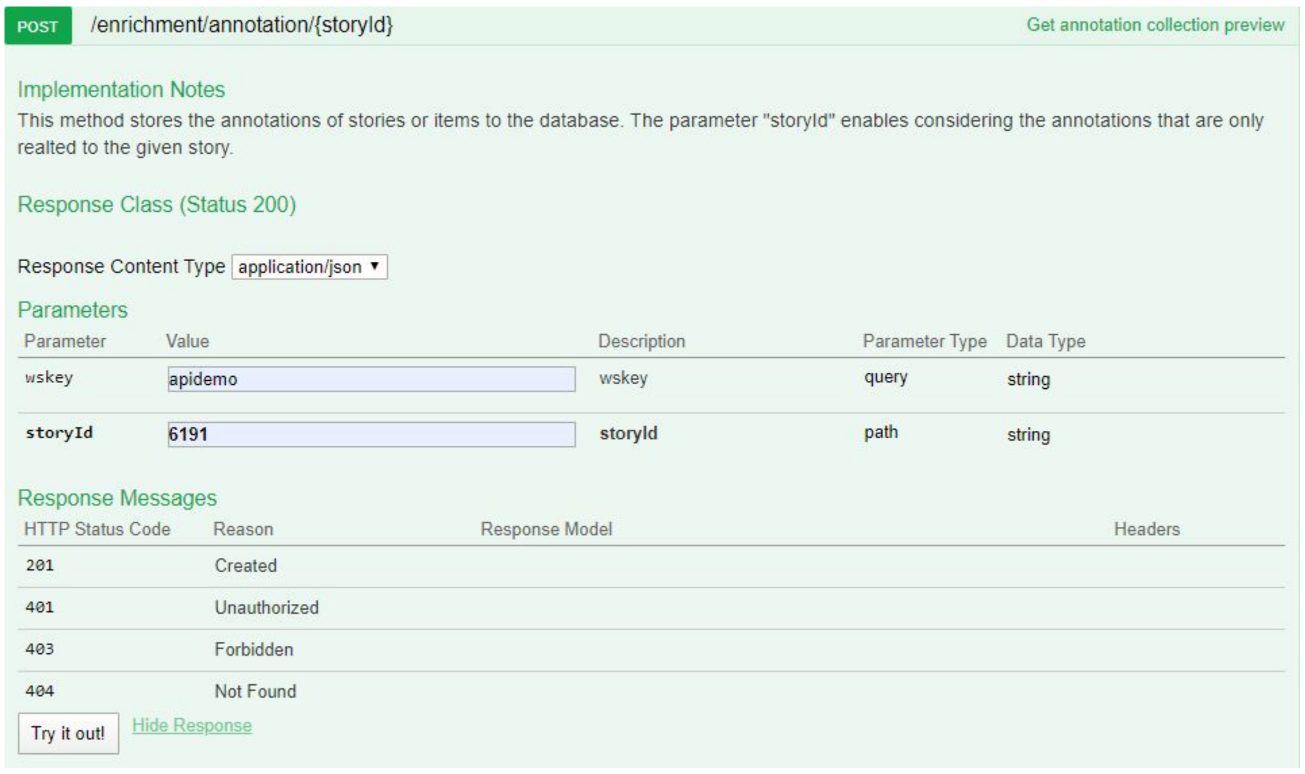
Co-financed by the Connecting Europe
Facility of the European Union

Enrichment Services

Computing semantic enrichments

The semantic enrichments using named entities (persons, locations or organizations) found in the stories or items elements are computed with the named entity recognition (NER) and linking services (see below). For each named entity discovered by the entity recognition service, the linking to linked data resources (e.g. Wikidata entities) is performed. The enrichments are then enhanced with some additional information forming the enrichments annotations that are stored in the database (using the given POST requests) and can be accessed from it (using the given GET requests).

Figure 10 shows the request to compute and save the semantic enrichments into the database, in a standardized representation as web annotations. The "storyId" parameter the document from Transcribathon Platform for which the annotations will be generated. If not performed yet, this request will also trigger the NER analysis for the given story.


POST /enrichment/annotation/{storyId}

Get annotation collection preview

Implementation Notes

This method stores the annotations of stories or items to the database. The parameter "storyId" enables considering the annotations that are only realted to the given story.

Response Class (Status 200)

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string

Response Messages

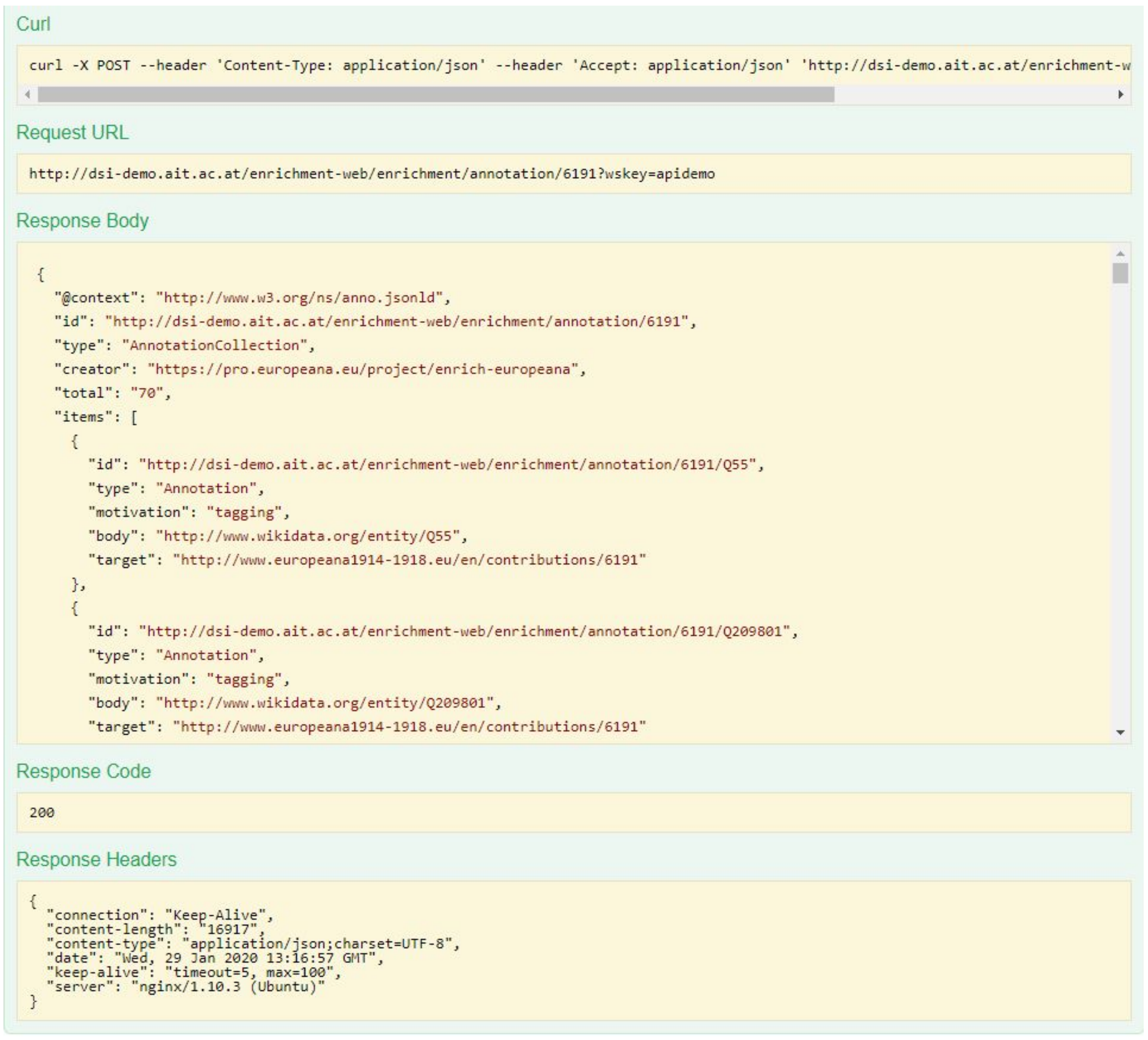
HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 10: Compute semantic enrichments for stories request

Figure 11 shows the request and response for the enrichments annotations for the story with storyId=10 (see Figure 10). The response represents a collection of annotations using the JSON-LD format following the W3C Web Annotation standard. Each individual enrichment is serialized according to the semantic tagging specifications within the Europeana Annotation API. The target resource represents the URI of the resource within the Transcribathon Platform. The body represents the Wikidata entity identifier which is linked to the found entity.

Co-financed by the Connecting Europe Facility of the European Union



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-w
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191",
  "type": "AnnotationCollection",
  "creator": "https://pro.europeana.eu/project/enrich-europeana",
  "total": "70",
  "items": [
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/Q55",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "http://www.wikidata.org/entity/Q55",
      "target": "http://www.europeana1914-1918.eu/en/contributions/6191"
    },
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/Q209801",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "http://www.wikidata.org/entity/Q209801",
      "target": "http://www.europeana1914-1918.eu/en/contributions/6191"
```

Response Code

```
200
```

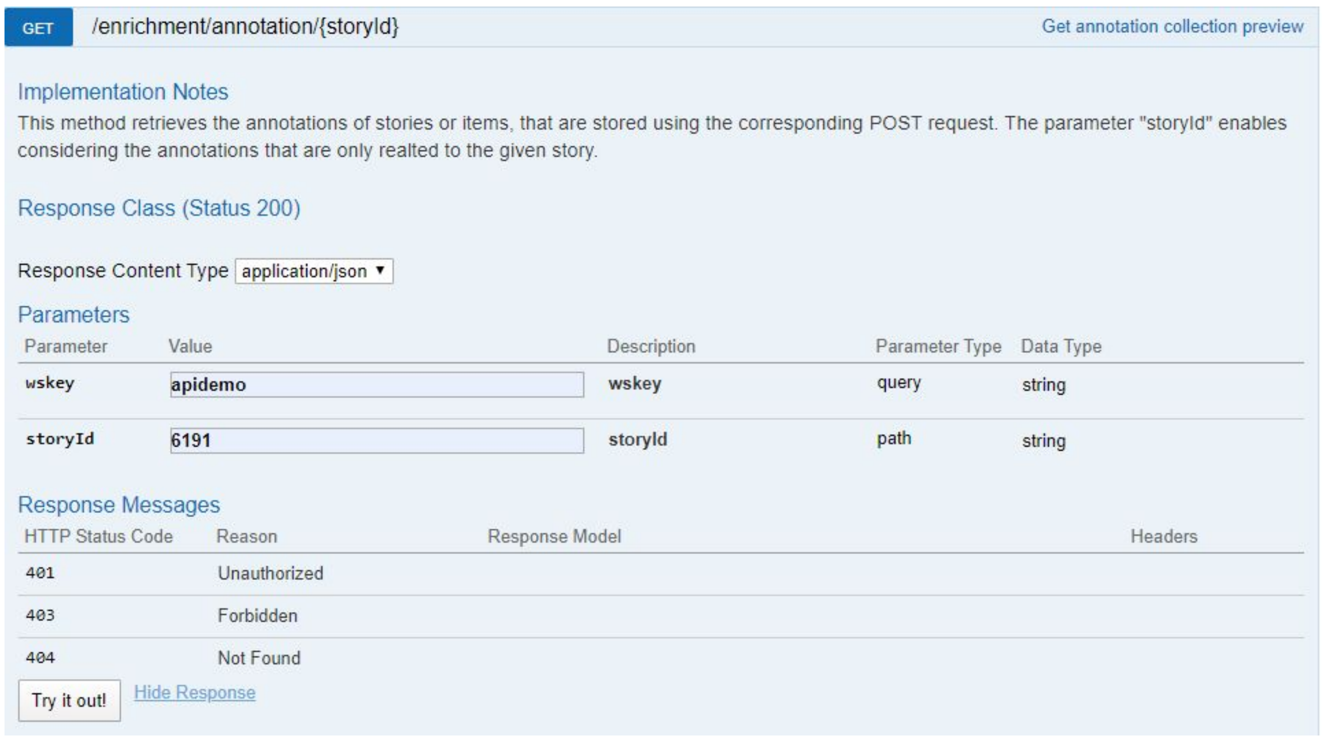
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "16917",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:16:57 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 11: Compute semantic enrichments for stories response

Retrieve enrichments request for stories

Figure 12 shows the invocation of the retrieve enrichments request through the swagger UI. In comparison to the above given POST request, this method only retrieves the enrichments annotations from the database. In case where no annotations are saved a corresponding message in the response is shown. Figure 13 represents the request and response for the input data from Figure 12. The data shown in the figure are the same as the one shown in Figure 11.



GET /enrichment/annotation/{storyId} Get annotation collection preview

Implementation Notes

This method retrieves the annotations of stories or items, that are stored using the corresponding POST request. The parameter "storyId" enables considering the annotations that are only realted to the given story.

Response Class (Status 200)

Response Content Type: application/json

Parameters

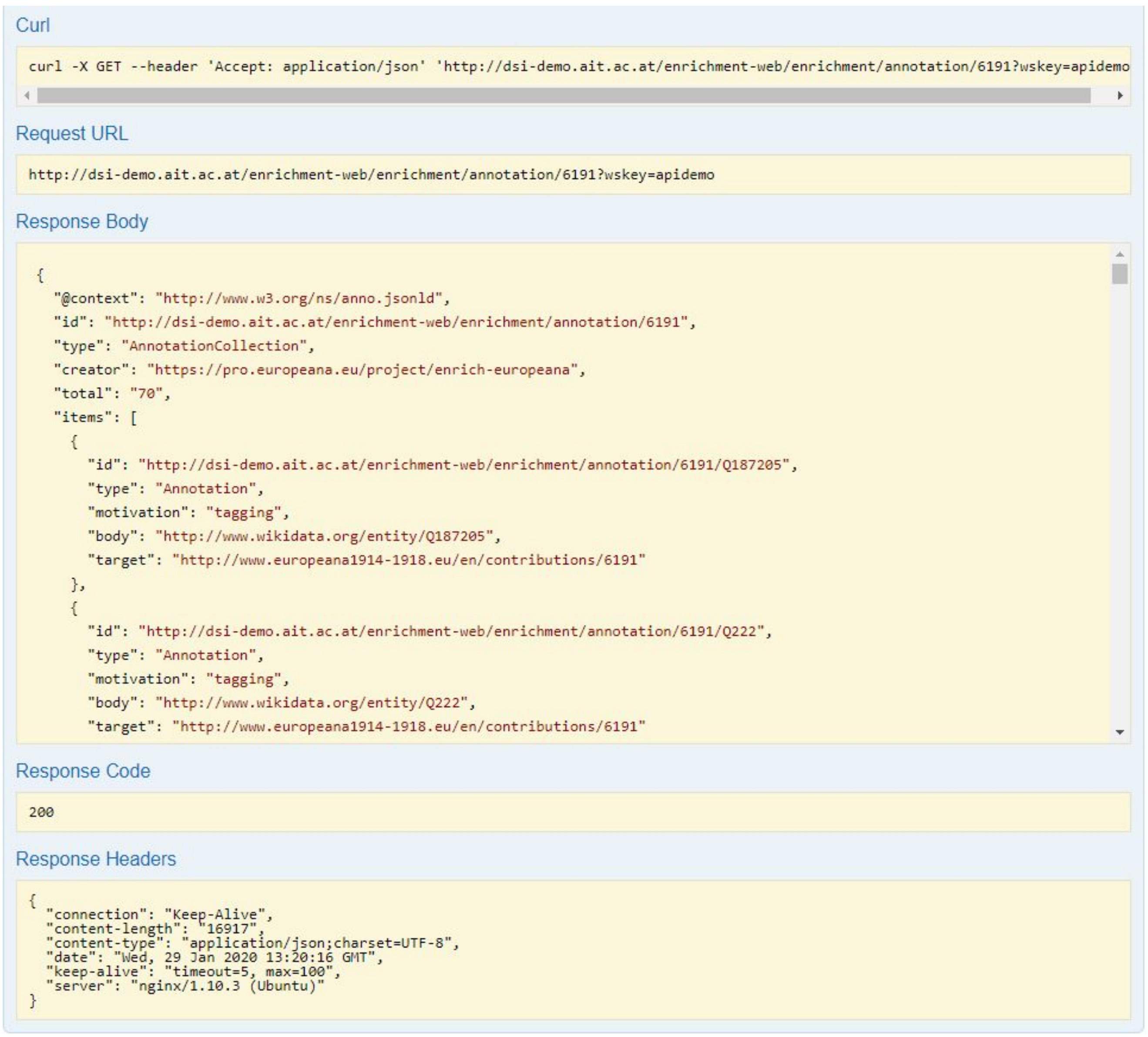
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 12:Retrieve semantic enrichments for stories request

Co-financed by the Connecting Europe Facility of the European Union



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191?wskey=apidemo
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191",
  "type": "AnnotationCollection",
  "creator": "https://pro.europeana.eu/project/enrich-europeana",
  "total": "70",
  "items": [
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/Q187205",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "http://www.wikidata.org/entity/Q187205",
      "target": "http://www.europeana1914-1918.eu/en/contributions/6191"
    },
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/Q222",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "http://www.wikidata.org/entity/Q222",
      "target": "http://www.europeana1914-1918.eu/en/contributions/6191"
```

Response Code

```
200
```

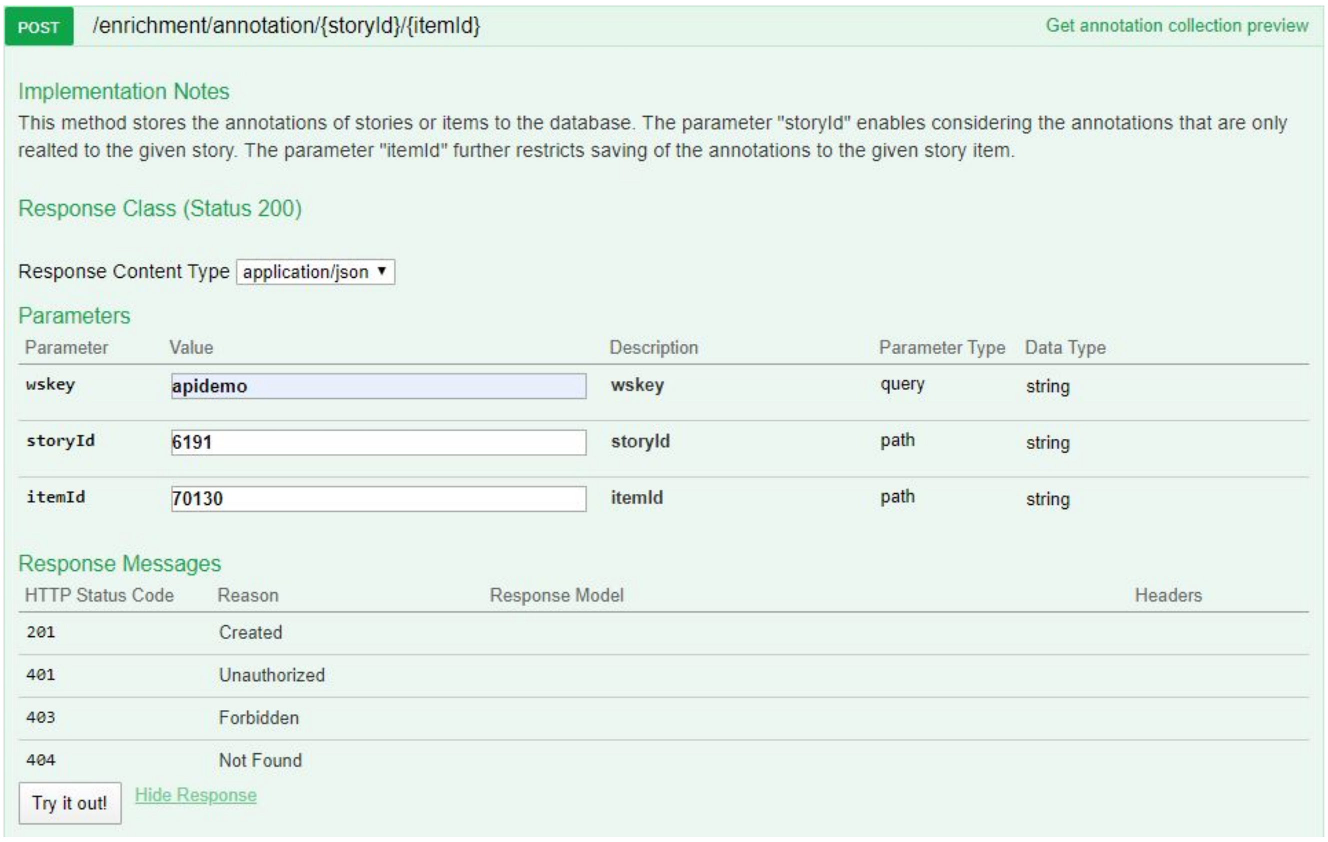
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "16917",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:20:16 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 13: Retrieve semantic enrichment for stories response

Compute enrichments request for items

The request to retrieve semantic enrichment for a given item through the Swagger UI is shown in Figure 14. The POST method enables saving the enrichments annotations for the given item to the database. An item is specified using the "storyId" (to which story an item belong) and "itemId" parameters. The behaviour of the method is basically the same as the above one for stories except that it works with items.



POST /enrichment/annotation/{storyId}/{itemId} Get annotation collection preview

Implementation Notes

This method stores the annotations of stories or items to the database. The parameter "storyId" enables considering the annotations that are only realted to the given story. The parameter "itemId" further restricts saving of the annotations to the given story item.

Response Class (Status 200)

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string
itemId	70130	itemId	path	string

Response Messages

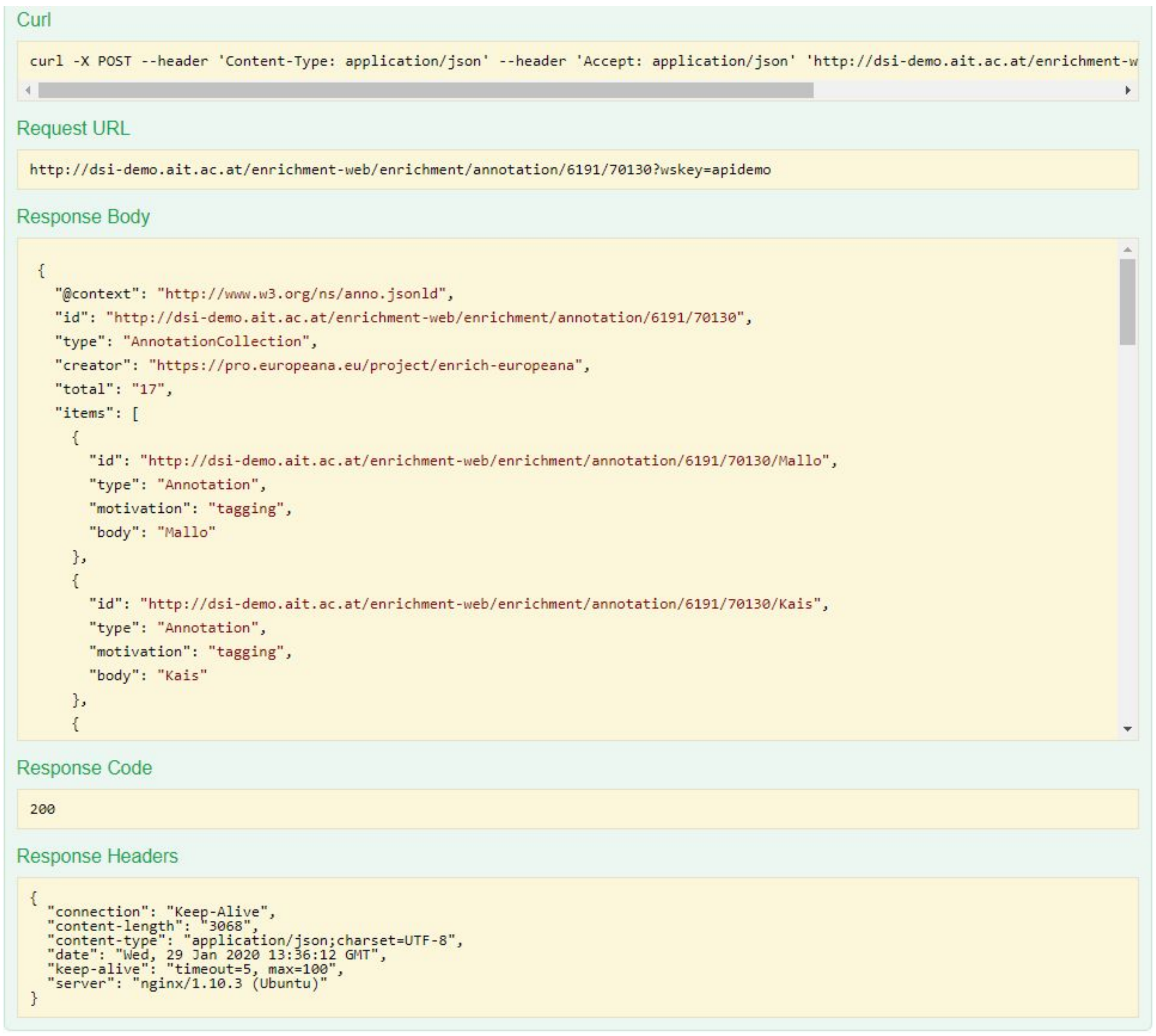
HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 14: Compute semantic enrichments for items request

Figure 15 shows the request and response of the enrichment service for the item with itemId=19676 (same as in Figure 14). Note the same JSON-LD format for the response as in the case of the previously shown enrichments in form of web annotations.

Co-financed by the Connecting Europe Facility of the European Union


Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-w
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130",
  "type": "AnnotationCollection",
  "creator": "https://pro.europeana.eu/project/enrich-europeana",
  "total": "17",
  "items": [
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130/Mallo",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "Mallo"
    },
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130/Kais",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "Kais"
    },
    {
```

Response Code

```
200
```

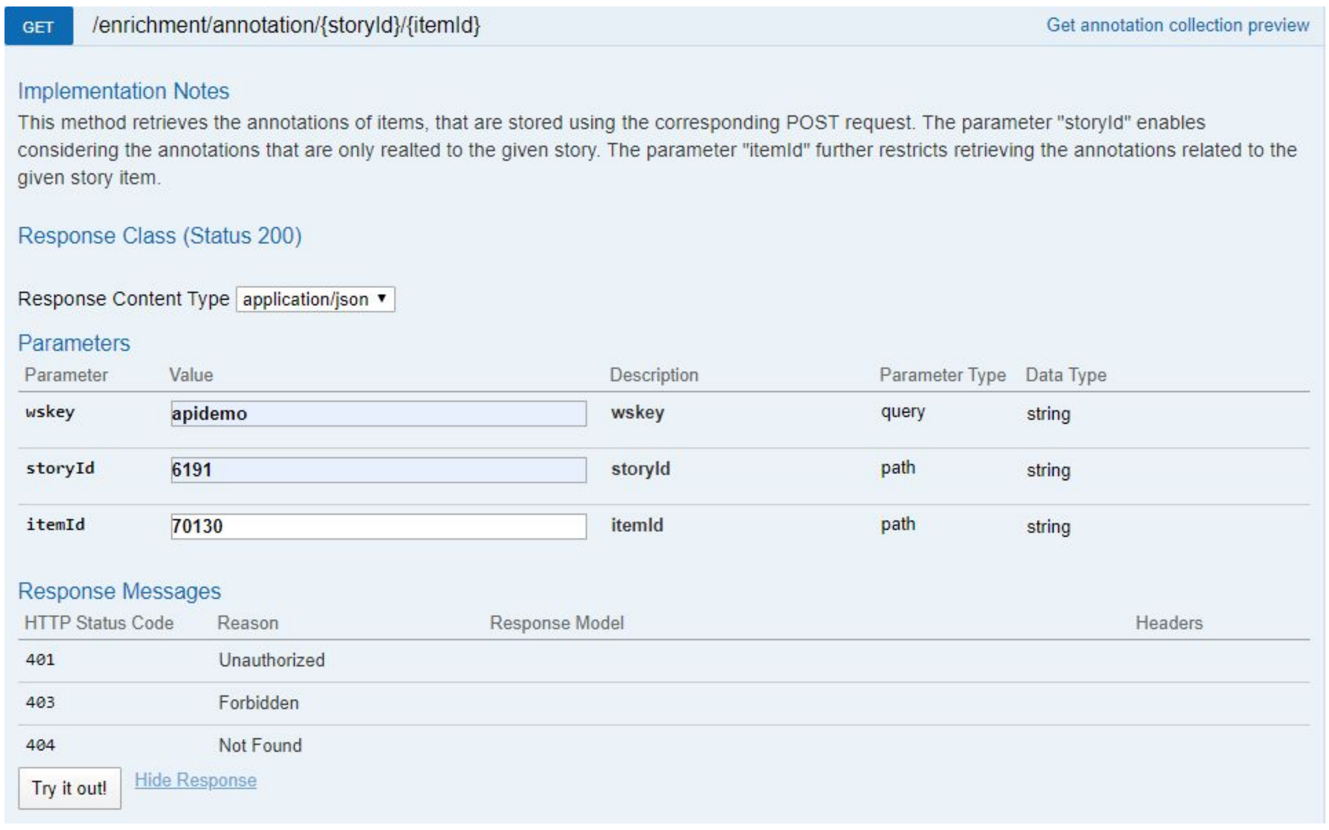
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "3068",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:36:12 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 15: Compute semantic enrichments for items response

Retrieve enrichments request for items

The service that retrieves enrichments for items (Figures 16 and 17) behaves in the same way as the above given requests for stories except it returns the enrichments for items. Please note that in this document we show all our POST and GET interfaces and their corresponding requests and responses. Even if some methods are intuitively clear, given the descriptions of the previously presented methods, we include the explicit presentation of all API methods, in order to have a complete documentation of the services.



GET /enrichment/annotation/{storyId}/{itemId} Get annotation collection preview

Implementation Notes

This method retrieves the annotations of items, that are stored using the corresponding POST request. The parameter "storyId" enables considering the annotations that are only realted to the given story. The parameter "itemId" further restricts retrieving the annotations related to the given story item.

Response Class (Status 200)

Response Content Type application/json

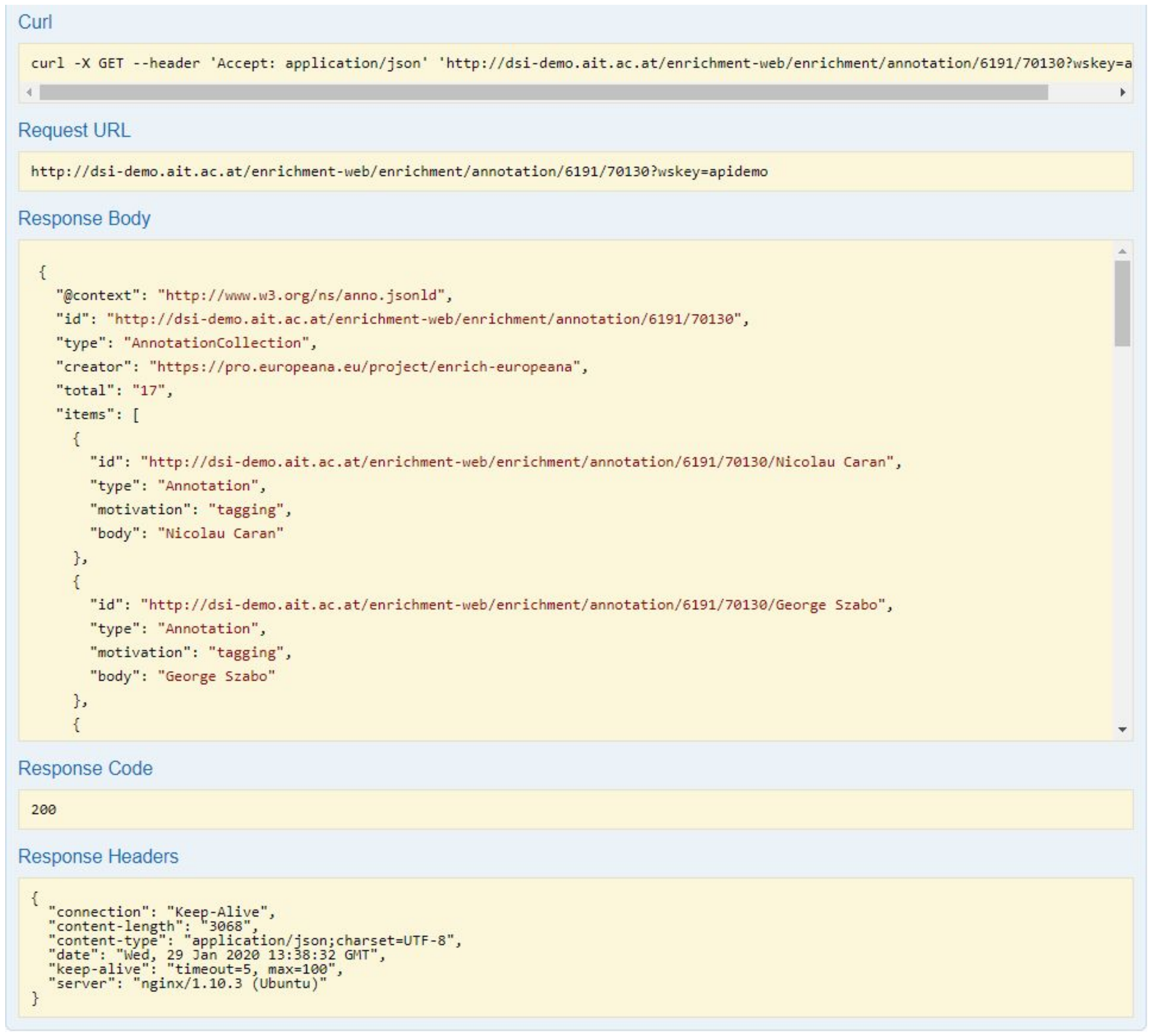
Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string
itemId	70130	itemId	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 16: Retrieve enrichments request for items request



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130?wskey=a
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130",
  "type": "AnnotationCollection",
  "creator": "https://pro.europeana.eu/project/enrich-europeana",
  "total": "17",
  "items": [
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130/Nicolau Caran",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "Nicolau Caran"
    },
    {
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130/George Szabo",
      "type": "Annotation",
      "motivation": "tagging",
      "body": "George Szabo"
    },
    {
```

Response Code

```
200
```

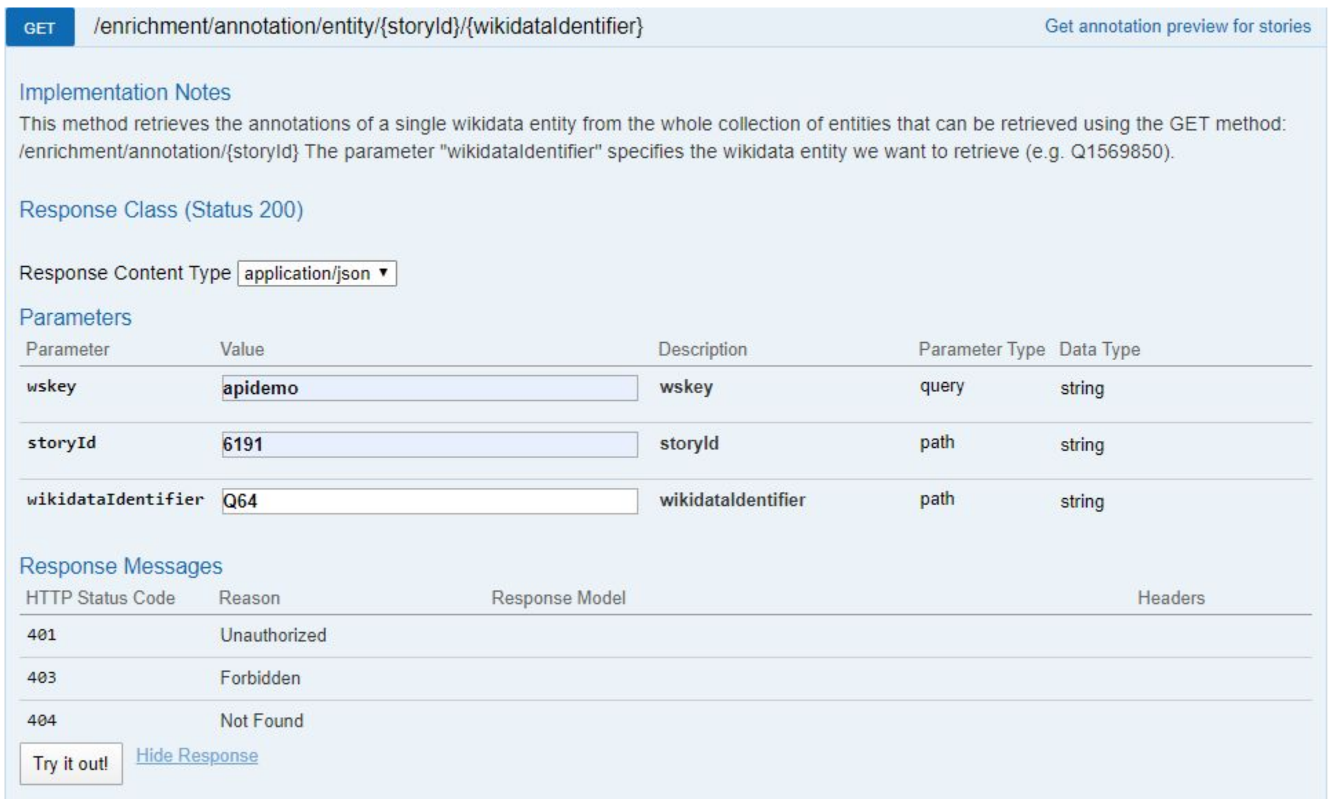
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "3068",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:38:32 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 17: Retrieve enrichments for items response

Retrieve individual enrichment for stories and items

Individual semantic enrichments for stories and items are retrieved by their URIs (Figures 18-21). In comparison to the semantic enrichments that deliver the collections of enrichment entities (Figures 10-17), the APIs for the individual semantic enrichments contain an additional parameter which is the identifier of the entity used for enrichment (i.e. wikidataIdentifier). Please have a look at the Wikidata model parameters (https://www.wikidata.org/wiki/Wikidata:Glossary) for more information about the notation (e.g. here Q64 represents the qualifier for Berlin). An example of the individual semantic enrichment retrieval is shown in Figures 19 and 21. If an invalid Wikidata identifier is provided, a corresponding message is shown.



GET /enrichment/annotation/entity/{storyId}/{wikidataIdentifier} Get annotation preview for stories

Implementation Notes

This method retrieves the annotations of a single wikidata entity from the whole collection of entities that can be retrieved using the GET method: /enrichment/annotation/{storyId} The parameter "wikidataIdentifier" specifies the wikidata entity we want to retrieve (e.g. Q1569850).

Response Class (Status 200)

Response Content Type application/json

Parameters

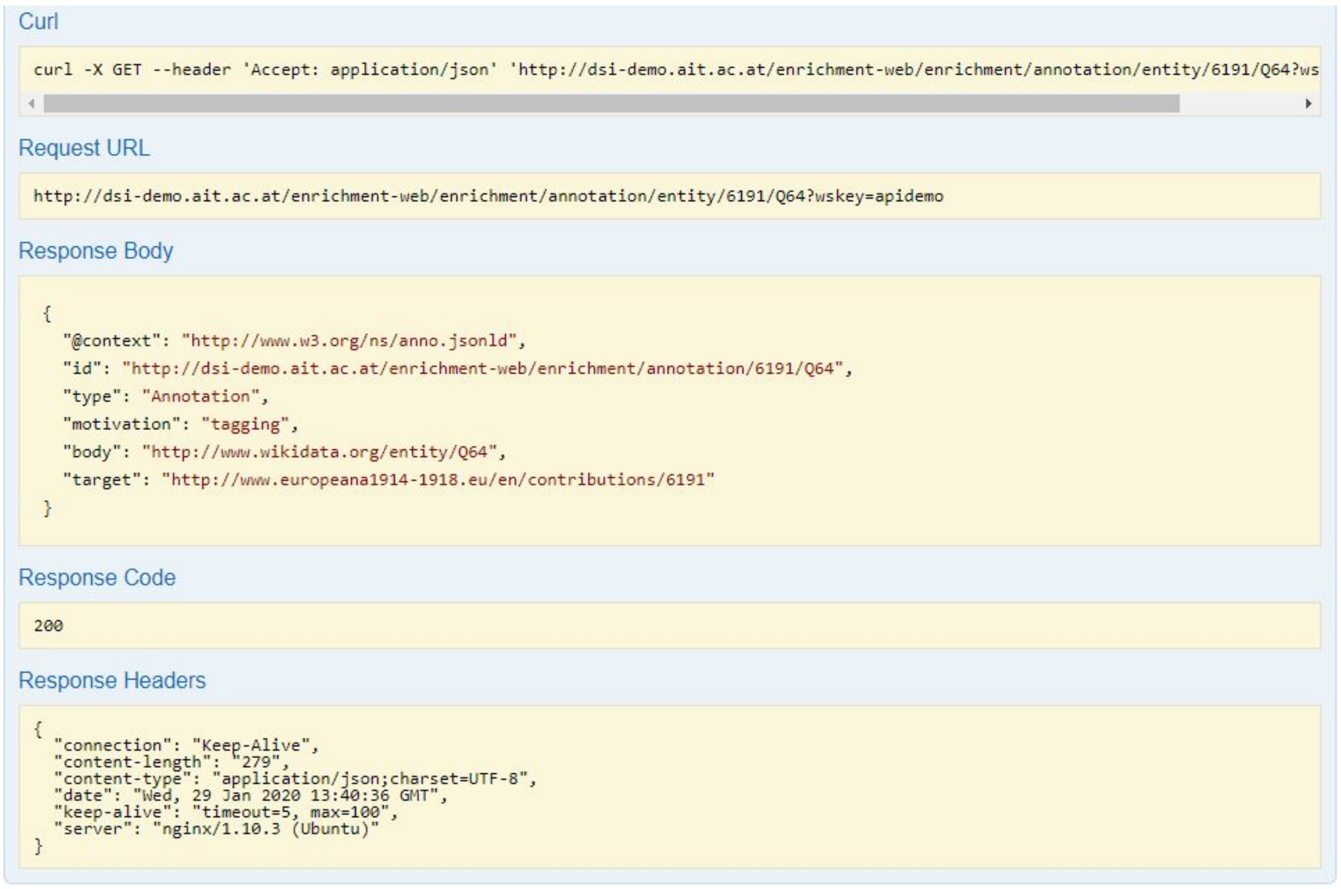
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string
wikidataIdentifier	Q64	wikidataIdentifier	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 18: Retrieve individual enrichment for stories request



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/entity/6191/Q64?ws
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/entity/6191/Q64?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/Q64",
  "type": "Annotation",
  "motivation": "tagging",
  "body": "http://www.wikidata.org/entity/Q64",
  "target": "http://www.europeana1914-1918.eu/en/contributions/6191"
}
```

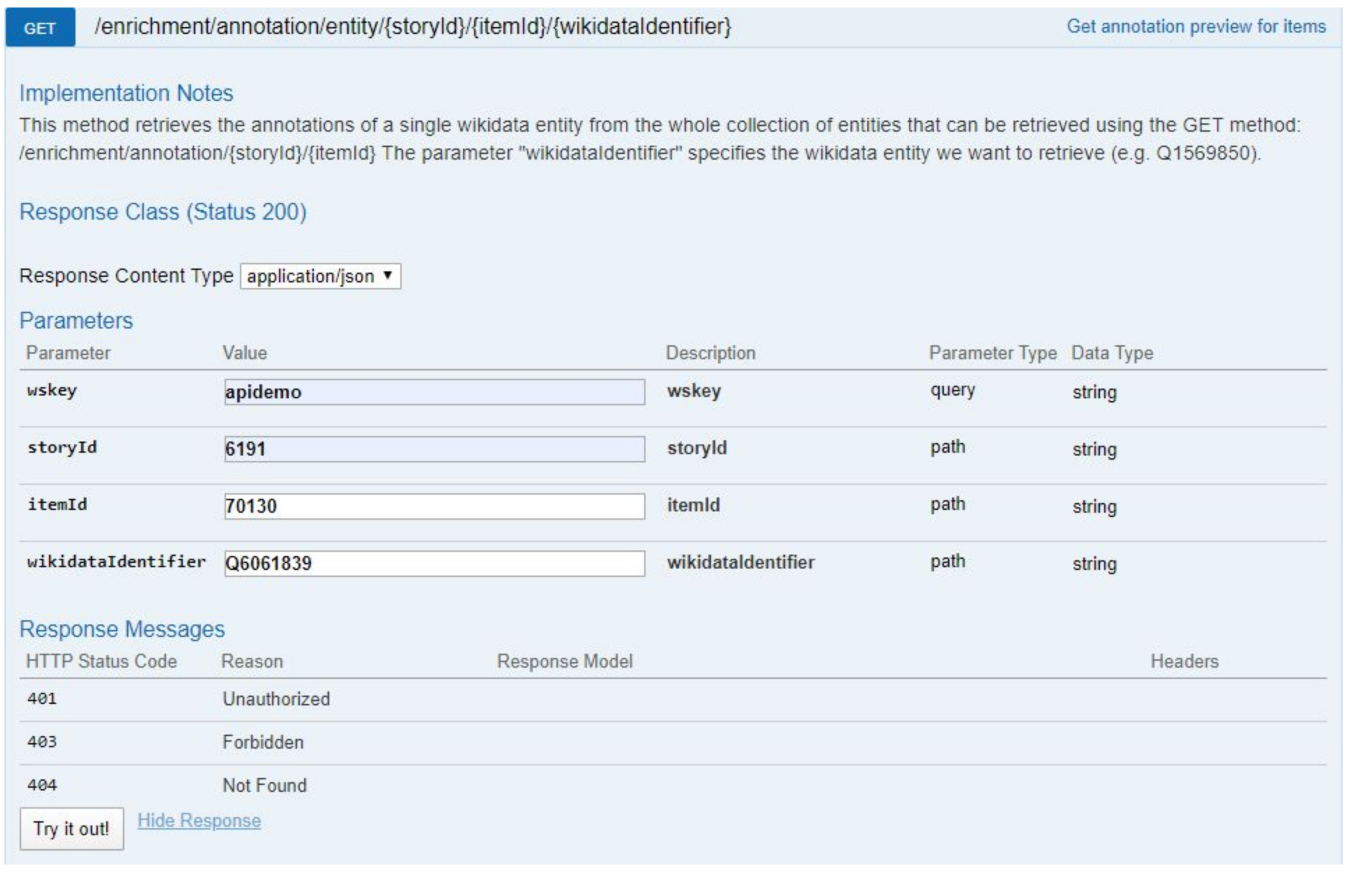
Response Code

```
200
```

Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "279",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:40:36 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 19: Retrieve individual enrichment for stories response



/enrichment/annotation/entity/{storyId}/{itemId}/{wikidataIdentifier}   Get annotation preview for items

Implementation Notes

This method retrieves the annotations of a single wikidata entity from the whole collection of entities that can be retrieved using the GET method: /enrichment/annotation/{storyId}/{itemId} The parameter "wikidataIdentifier" specifies the wikidata entity we want to retrieve (e.g. Q1569850).

Response Class (Status 200)

Response Content Type application/json

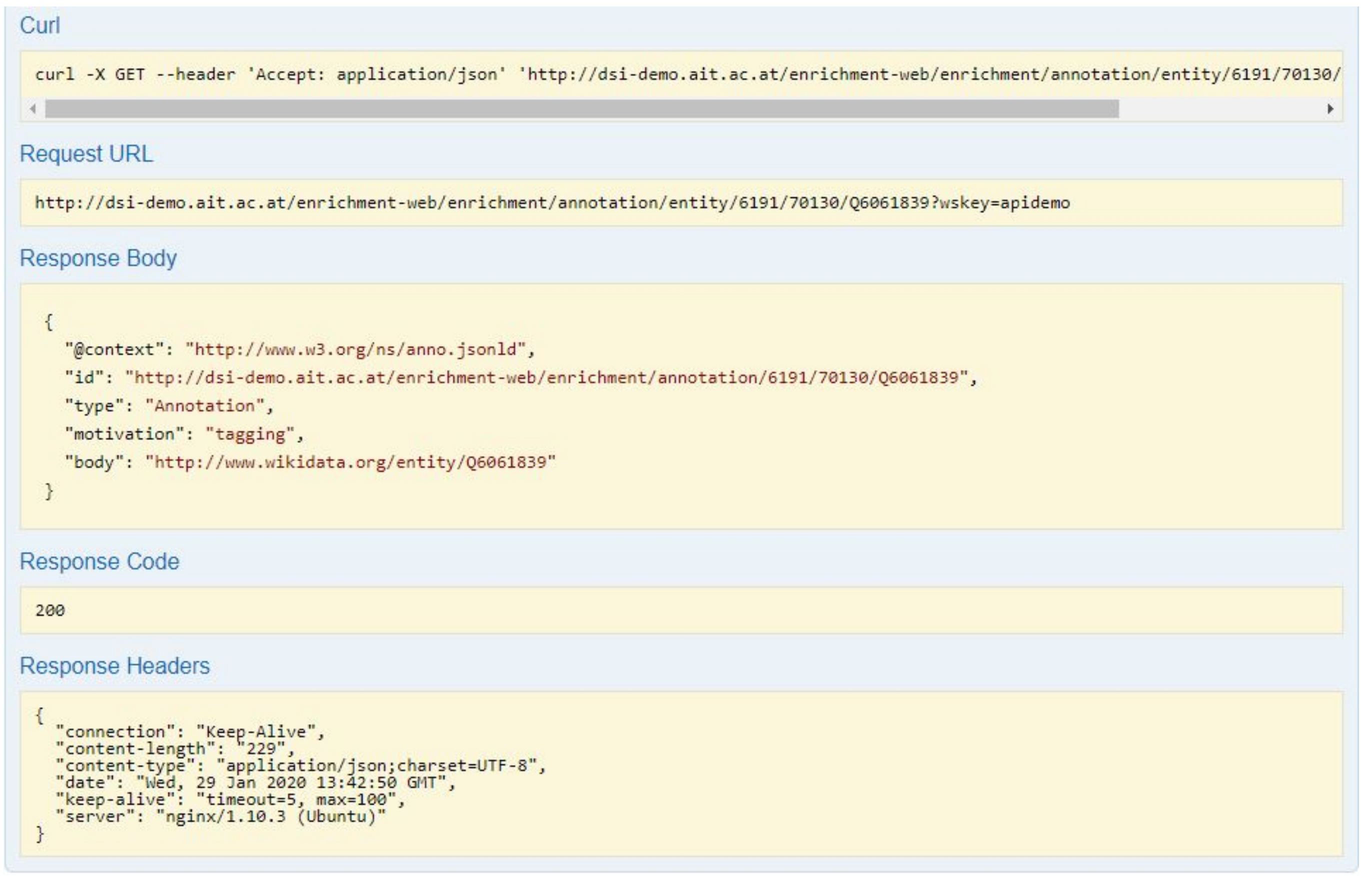
Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	6191	storyId	path	string
itemId	70130	itemId	path	string
wikidataIdentifier	Q6061839	wikidataIdentifier	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 20: Retrieve individual enrichment for items request


Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/entity/6191/70130/
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/entity/6191/70130/Q6061839?wskey=apidemo
```

Response Body

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/6191/70130/Q6061839",
  "type": "Annotation",
  "motivation": "tagging",
  "body": "http://www.wikidata.org/entity/Q6061839"
}
```

Response Code

```
200
```

Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "229",
  "content-type": "application/json;charset=UTF-8",
  "date": "Wed, 29 Jan 2020 13:42:50 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 21: Retrieve individual enrichment for items response

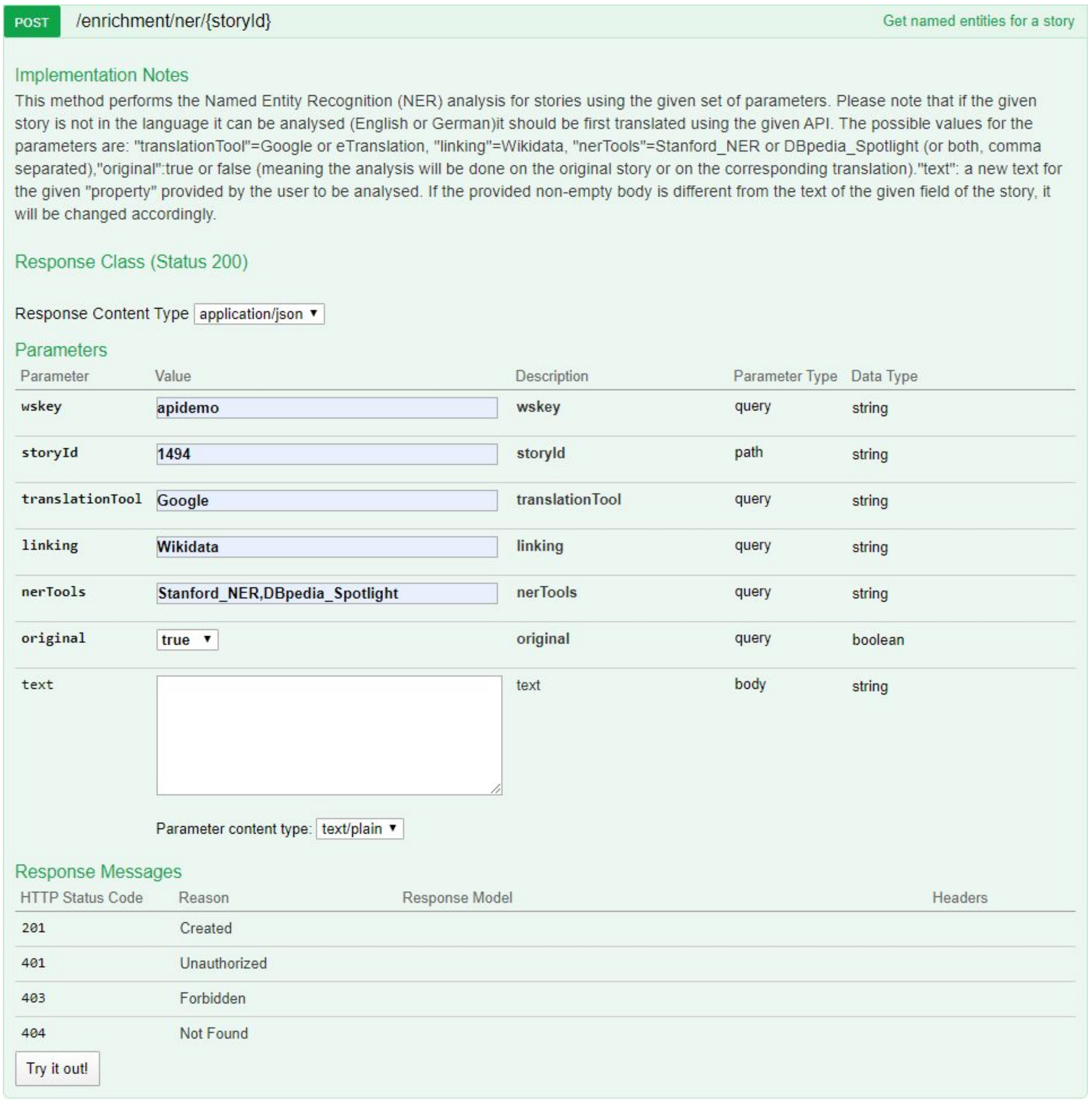
Named Entity Recognition for stories

The named entity recognition and linking functionality represents the core of Enrichment Services. Basically, named entity recognition is related to extracting the entities like persons, locations, and organisations from the given texts representing stories and/or items or their parts. The NER service is implemented using Stanford NER and DBpedia Spotlight services which represent the famous NER tools used today. The NER process can be applied directly on the text for some languages like English or German, for which there exist the models for pursuing the NER analysis while for the texts in other languages, the required machine translation needs to be performed. Beside the recognition of named entities, this service is also performing the linking of the found entities with resources from semantic web repositories like Wikidata.

Figure 22 shows the swagger console UI used to build requests sent for performing named entity recognition and linking. Please have a look at the Implementation Notes part in the figure for the explanation of the parameters used in the request. For the "nerTools" parameter when both tools (Stanford_NER and DBpedia_Spotlight) are applied together, the results will include the union of the

Co-financed by the Connecting Europe Facility of the European Union

results obtained when the tools are applied separately. The linking to DBpedia resources is performed automatically by Spotlight. Through the "linking" parameter the user has the possibility to resolve found entities against Wikidata and Europeana repositories. If any of the provided values for the parameters is invalid, an HTTP Bad Request exception is shown with a corresponding message for the user.



POST /enrichment/ner/{storyId} — Get named entities for a story

Implementation Notes

This method performs the Named Entity Recognition (NER) analysis for stories using the given set of parameters. Please note that if the given story is not in the language it can be analysed (English or German)it should be first translated using the given API. The possible values for the parameters are: "translationTool"=Google or eTranslation, "linking"=Wikidata, "nerTools"=Stanford_NER or DBpedia_Spotlight (or both, comma separated),"original":true or false (meaning the analysis will be done on the original story or on the corresponding translation)."text": a new text for the given "property" provided by the user to be analysed. If the provided non-empty body is different from the text of the given field of the story, it will be changed accordingly.

Response Class (Status 200)

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
translationTool	Google	translationTool	query	string
linking	Wikidata	linking	query	string
nerTools	Stanford_NER,DBpedia_Spotlight	nerTools	query	string
original	true	original	query	boolean
text		text	body	string

Parameter content type: text/plain

Response Messages

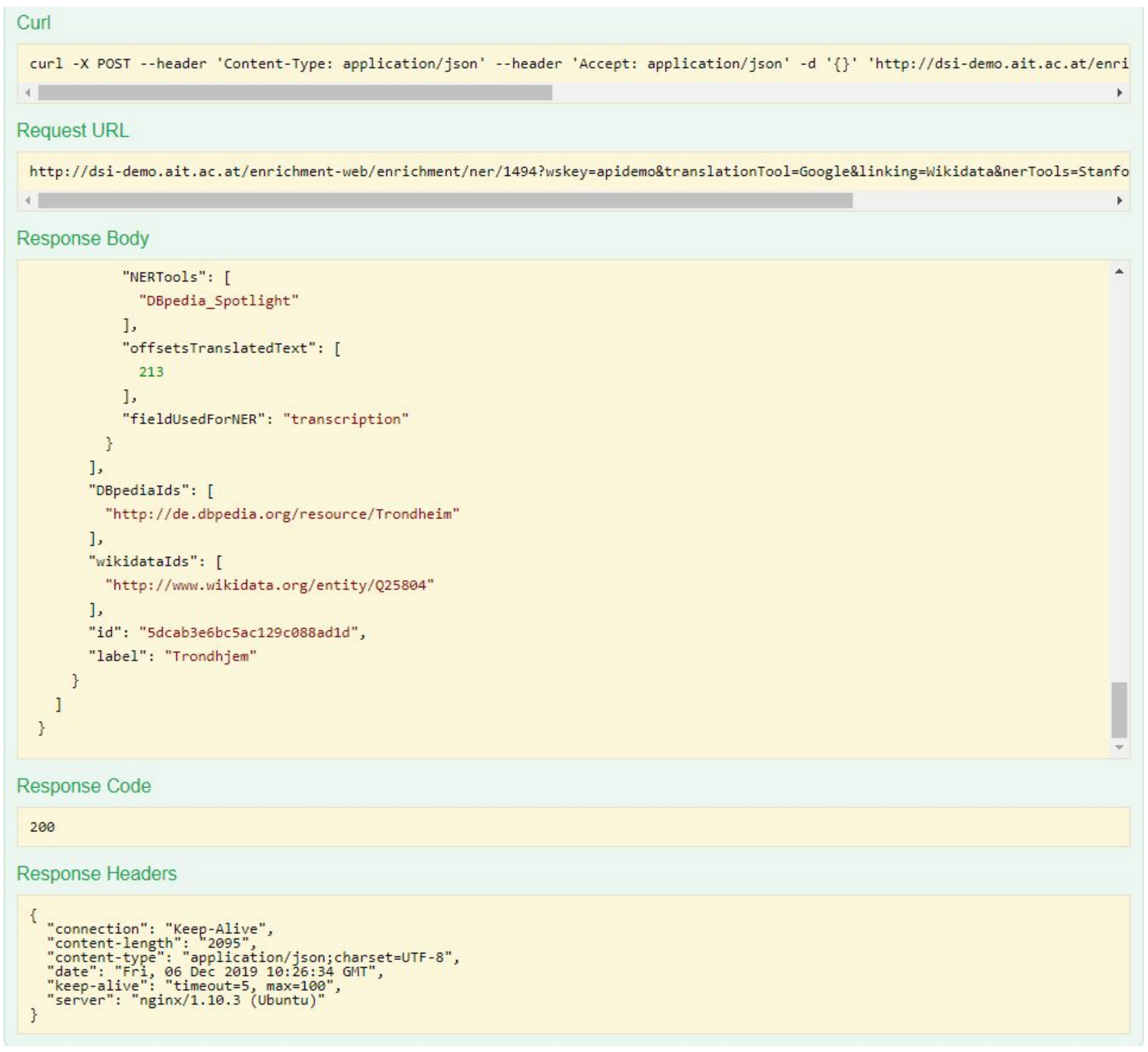
HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Fig 22: Request to perform Named Entity Recognition for stories

Figure 23 shows the response of the NER service with the request data presented in Figure 22. In the part of the response shown, we see the Trondhjem entity identified by Spotlight (i.e. "DBpediaIds" field) and also by Stanford_NER (i.e. "wikidataIds"). The field "preferredWikidataIds" (not visible in the figure) is based on identified DBpedia resource where the corresponding Wikidata entity is returned. The position of the found entities in the analysed text is saved in the "positionEntities" field (see e.g. Figure 25). In the provided example, the Trondhjem city was found in the english translation of the document starting at the 213st character.

Co-financed by the Connecting Europe
Facility of the European Union



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{}' 'http://dsi-demo.ait.ac.at/enri
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494?wskey=apidemo&translationTool=Google&linking=Wikidata&nerTools=Stanfo
```

Response Body

```
            "NERTools": [
              "DBpedia_Spotlight"
            ],
            "offsetsTranslatedText": [
              213
            ],
            "fieldUsedForNER": "transcription"
          }
        ],
        "DBpediaIds": [
          "http://de.dbpedia.org/resource/Trondheim"
        ],
        "wikidataIds": [
          "http://www.wikidata.org/entity/Q25804"
        ],
        "id": "5dcab3e6bc5ac129c088ad1d",
        "label": "Trondhjem"
      }
    ]
  }
```

Response Code

```
200
```

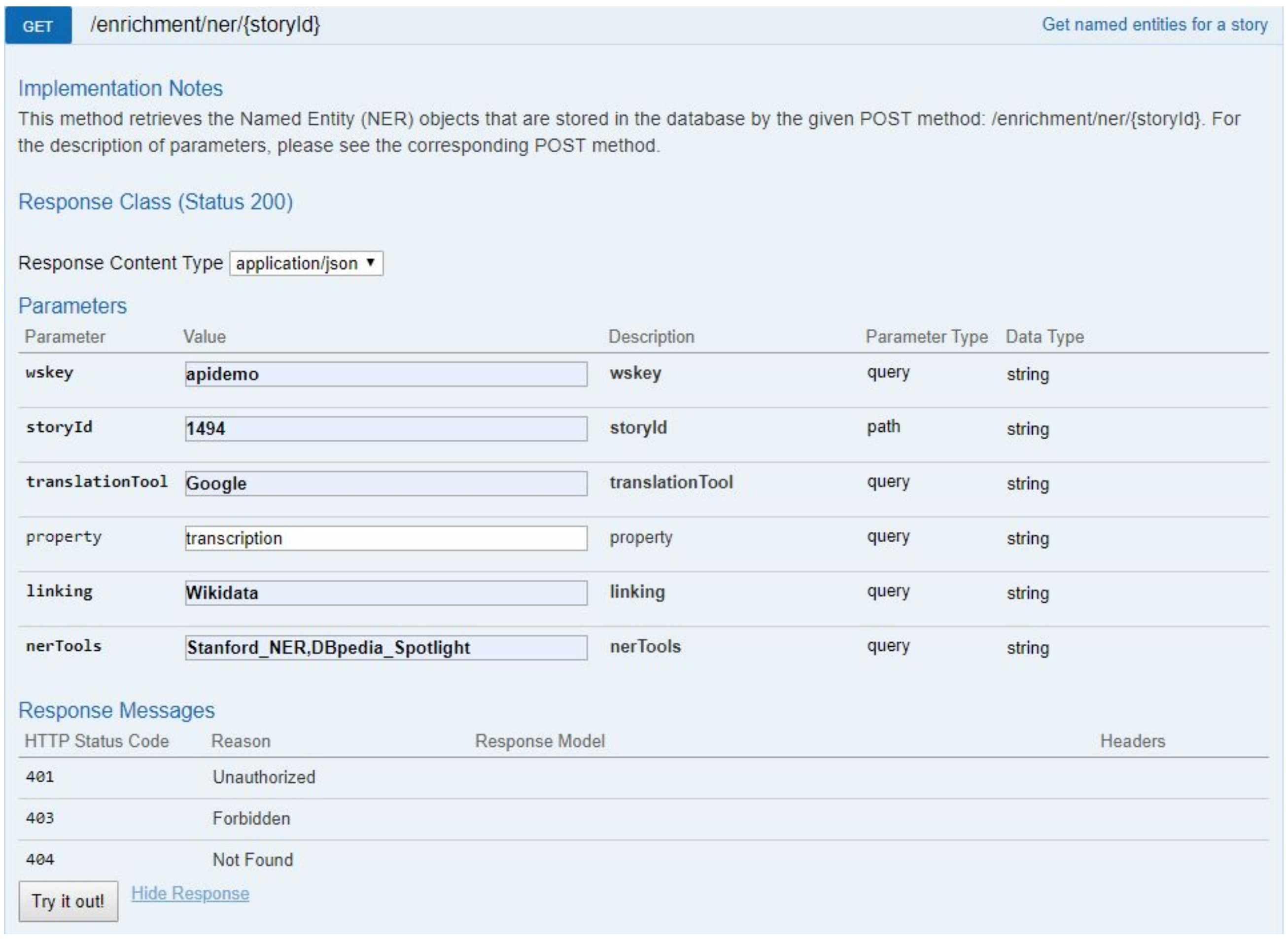
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "2095",
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 10:26:34 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 23: Response of the Named Entity Recognition for stories

Retrieve identified Named Entities for stories

After pursuing the NER analysis with the given POST method, the results are saved in the database and can be retrieved using the given GET method (shown in Figure 24). The parameters used in the GET method are the same as the one provided for the POST method.



GET /enrichment/ner/{storyId} Get named entities for a story

Implementation Notes

This method retrieves the Named Entity (NER) objects that are stored in the database by the given POST method: /enrichment/ner/{storyId}. For the description of parameters, please see the corresponding POST method.

Response Class (Status 200)

Response Content Type application/json

Parameters

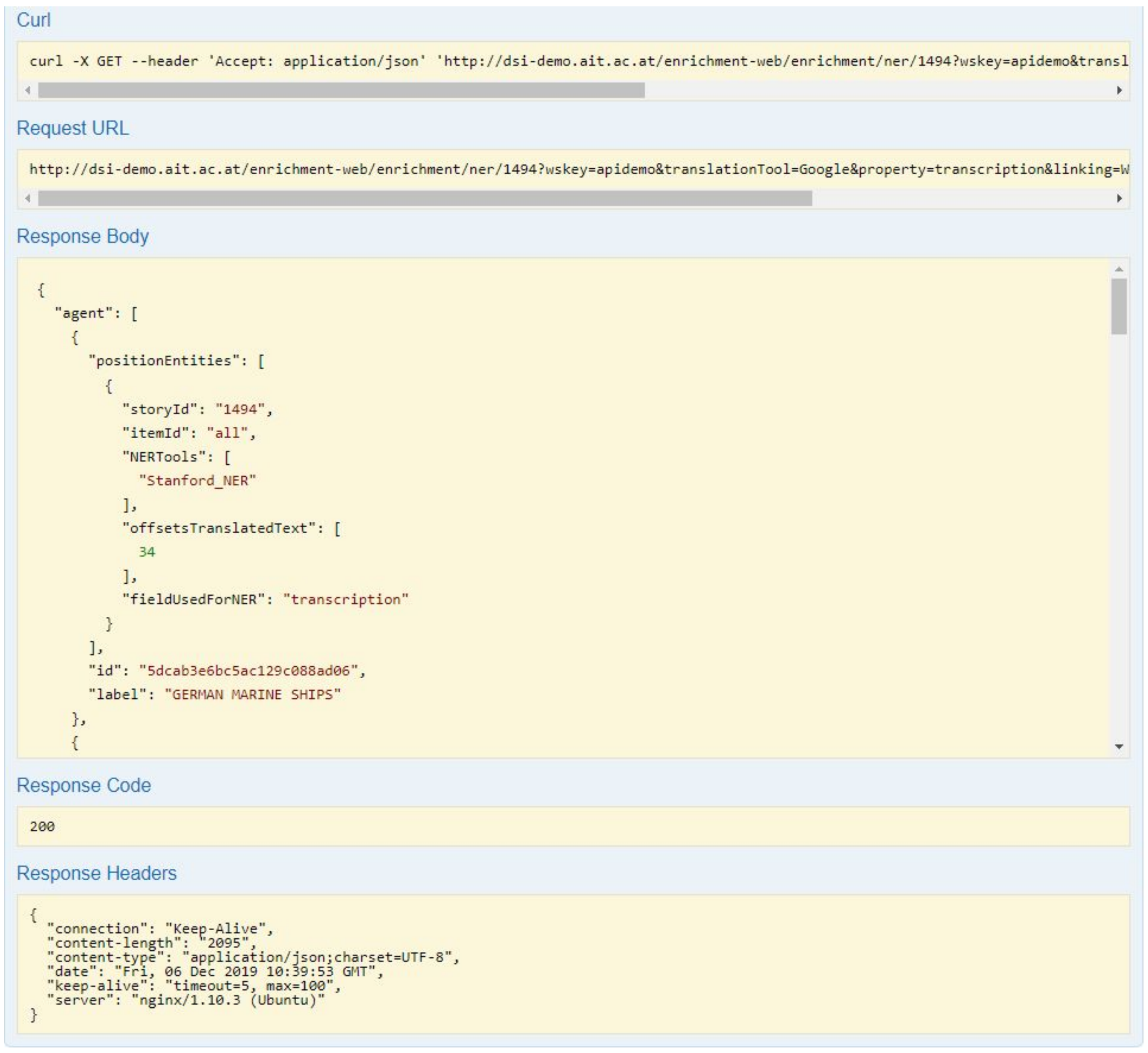
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
translationTool	Google	translationTool	query	string
property	transcription	property	query	string
linking	Wikidata	linking	query	string
nerTools	Stanford_NER,DBpedia_Spotlight	nerTools	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 24: Retrieve identified Named Entities for stories

Co-financed by the Connecting Europe Facility of the European Union



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494?wskey=apidemo&transl
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494?wskey=apidemo&translationTool=Google&property=transcription&linking=W
```

Response Body

```
{
  "agent": [
    {
      "positionEntities": [
        {
          "storyId": "1494",
          "itemId": "all",
          "NERTools": [
            "Stanford_NER"
          ],
          "offsetsTranslatedText": [
            34
          ],
          "fieldUsedForNER": "transcription"
        }
      ],
      "id": "5dcab3e6bc5ac129c088ad06",
      "label": "GERMAN MARINE SHIPS"
    },
    {
```

Response Code

```
200
```

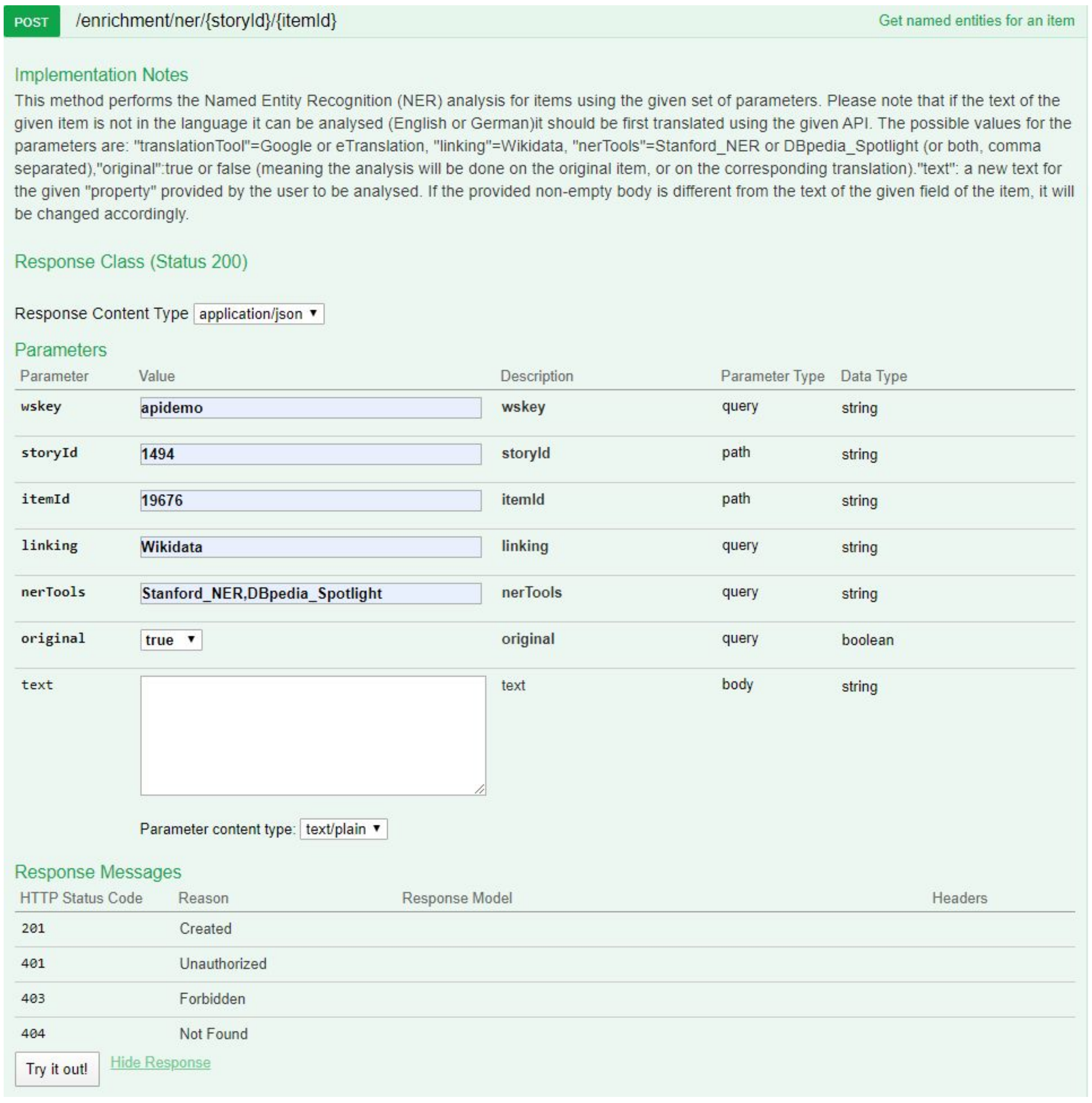
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "2095",
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 10:39:53 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 25: Retrieve Named Entities response for stories

Named Entity Recognition for items

Figures 26 and 27 shows the NER services for items which behave in the same way as those for stories except the analysis is done on the level of items. Consequently, the only difference is represented by the parameter indicating the individual item in a story (i.e. itemId). The request parameters have the same meaning and format response uses the same data model.


POST /enrichment/ner/{storyId}/{itemId} Get named entities for an item

Implementation Notes

This method performs the Named Entity Recognition (NER) analysis for items using the given set of parameters. Please note that if the text of the given item is not in the language it can be analysed (English or German)it should be first translated using the given API. The possible values for the parameters are: "translationTool"=Google or eTranslation, "linking"=Wikidata, "nerTools"=Stanford_NER or DBpedia_Spotlight (or both, comma separated),"original":true or false (meaning the analysis will be done on the original item, or on the corresponding translation)."text": a new text for the given "property" provided by the user to be analysed. If the provided non-empty body is different from the text of the given field of the item, it will be changed accordingly.

Response Class (Status 200)

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
itemId	19676	itemId	path	string
linking	Wikidata	linking	query	string
nerTools	Stanford_NER,DBpedia_Spotlight	nerTools	query	string
original	true	original	query	boolean
text		text	body	string

Parameter content type: text/plain

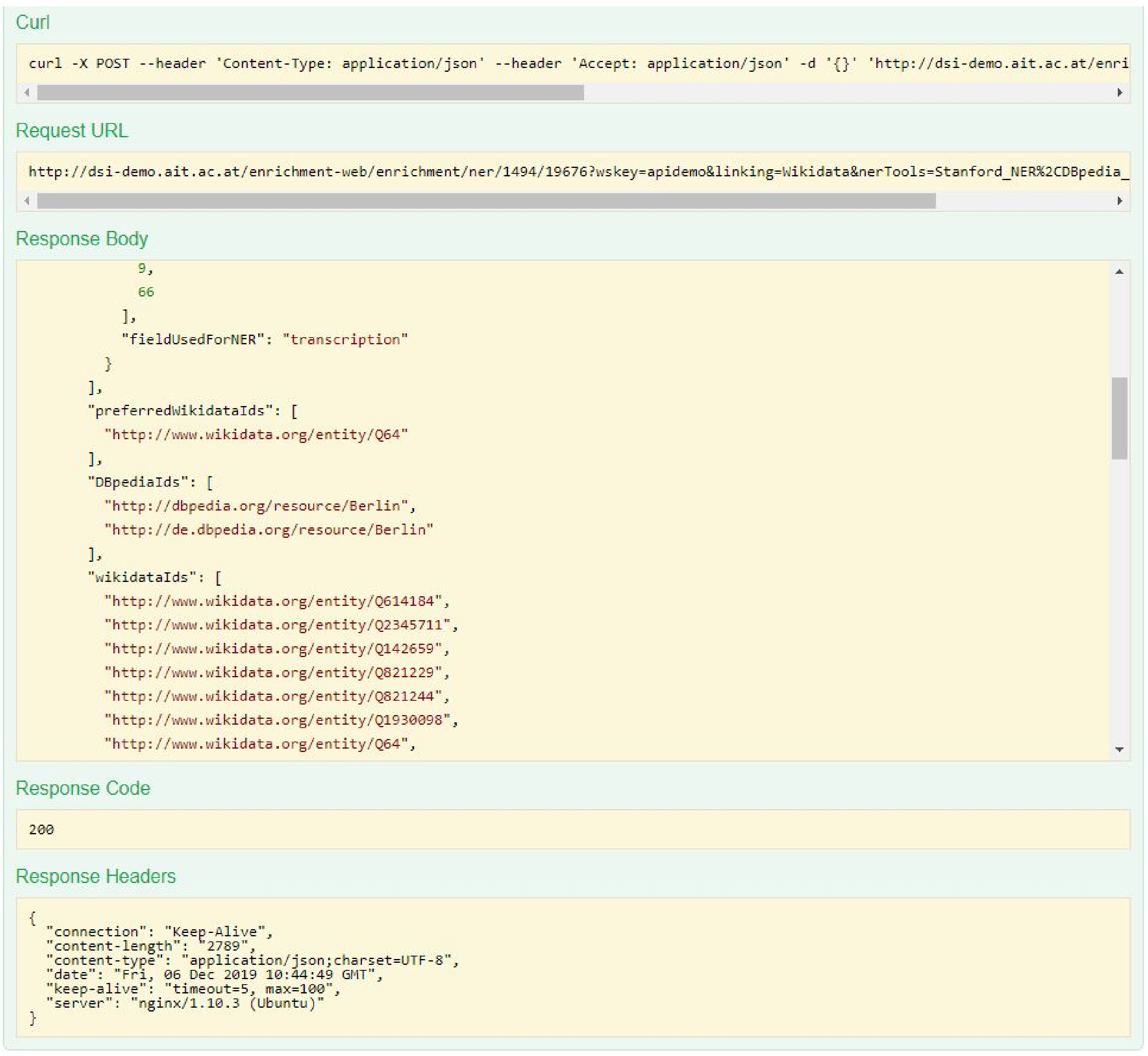
Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 26: Request to perform Named Entity Recognition for items

Co-financed by the Connecting Europe
Facility of the European Union


Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{}' 'http://dsi-demo.ait.ac.at/enri
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494/19676?wskey=apidemo&linking=Wikidata&nerTools=Stanford_NER%2CDBpedia_
```

Response Body

```
              9,
              66
            ],
            "fieldUsedForNER": "transcription"
          }
        ],
        "preferredWikidataIds": [
          "http://www.wikidata.org/entity/Q64"
        ],
        "DBpediaIds": [
          "http://dbpedia.org/resource/Berlin",
          "http://de.dbpedia.org/resource/Berlin"
        ],
        "wikidataIds": [
          "http://www.wikidata.org/entity/Q614184",
          "http://www.wikidata.org/entity/Q2345711",
          "http://www.wikidata.org/entity/Q142659",
          "http://www.wikidata.org/entity/Q821229",
          "http://www.wikidata.org/entity/Q821244",
          "http://www.wikidata.org/entity/Q1930098",
          "http://www.wikidata.org/entity/Q64",
```

Response Code

```
200
```

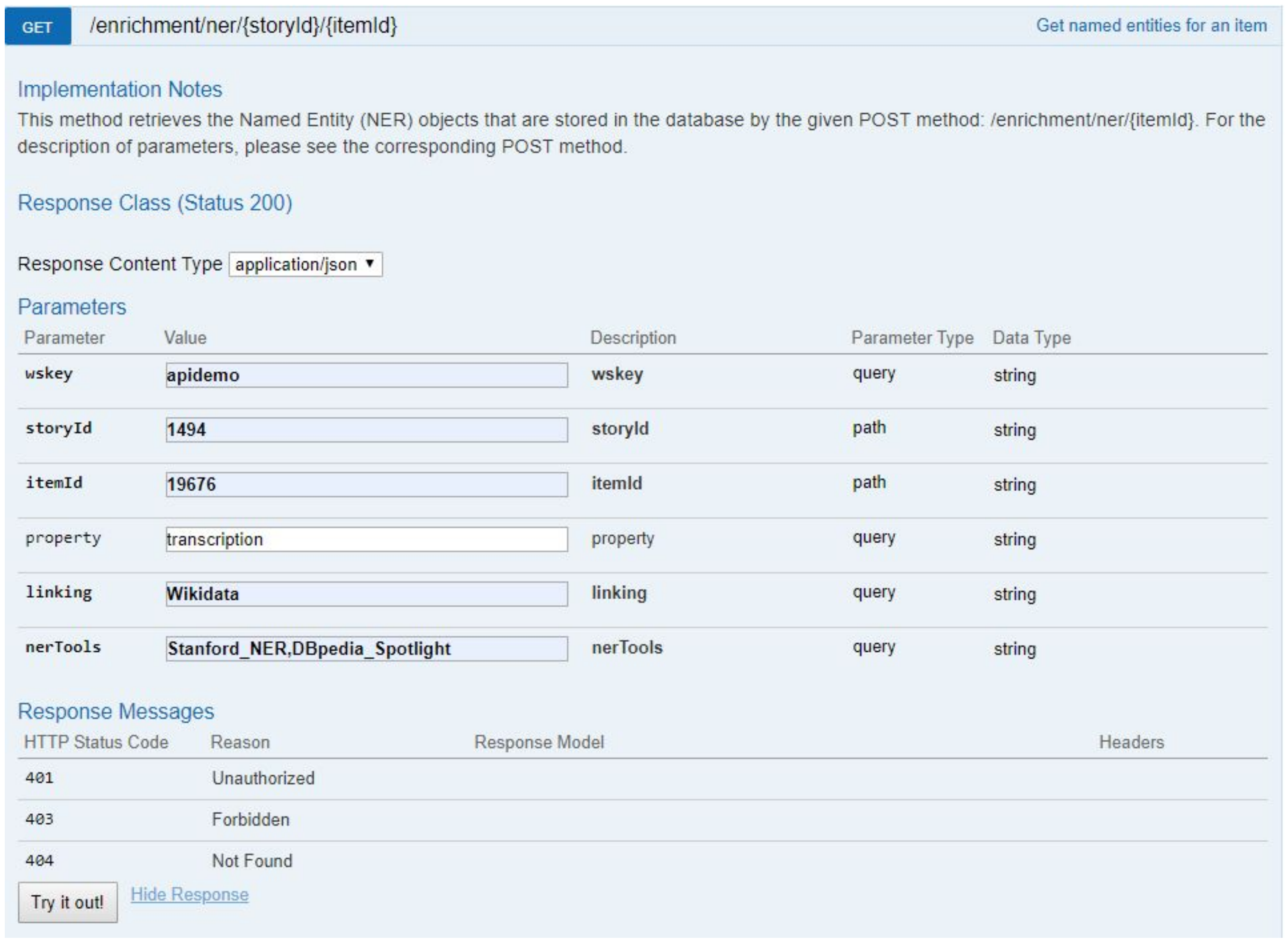
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "2789",
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 10:44:49 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 27: Response of the Named Entity Recognition for items

Enrichment Named Entity Recognition GET request for items

After performing the computation of automatic enrichments, these can be retrieved multiple times from the database without the need to perform the natural language processing operations. Figure 28 presents the request for retrieving previously identified named entities and Figure 29 presents the coresponding response.


GET /enrichment/ner/{storyId}/{itemId} — Get named entities for an item

Implementation Notes

This method retrieves the Named Entity (NER) objects that are stored in the database by the given POST method: /enrichment/ner/{itemId}. For the description of parameters, please see the corresponding POST method.

Response Class (Status 200)

Response Content Type: application/json

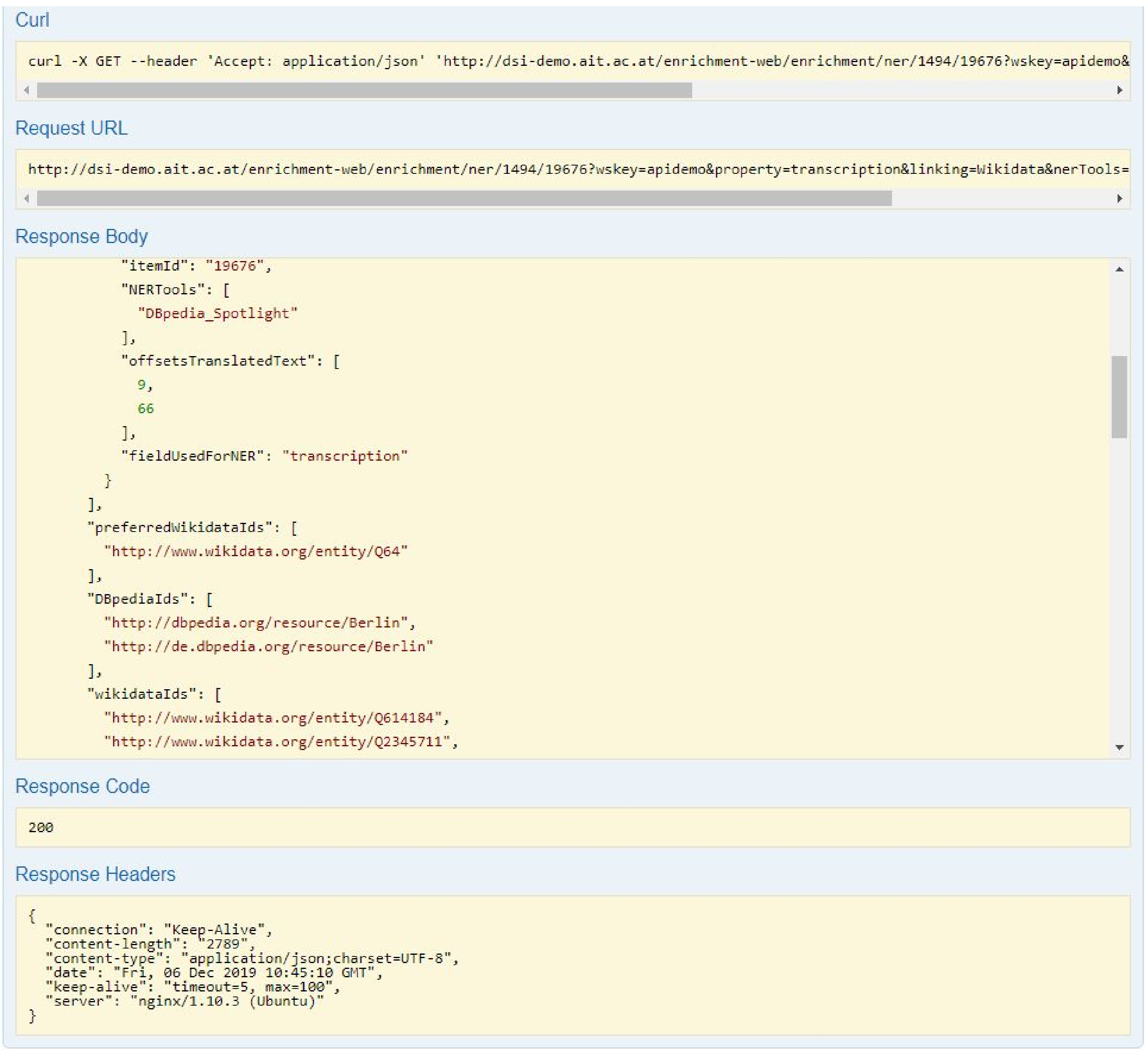
Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
storyId	1494	storyId	path	string
itemId	19676	itemId	path	string
property	transcription	property	query	string
linking	Wikidata	linking	query	string
nerTools	Stanford_NER,DBpedia_Spotlight	nerTools	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 28: Retrieve identified Named Entities for items



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494/19676?wskey=apidemo&
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/ner/1494/19676?wskey=apidemo&property=transcription&linking=Wikidata&nerTools=
```

Response Body

```
            "itemId": "19676",
            "NERTools": [
              "DBpedia_Spotlight"
            ],
            "offsetsTranslatedText": [
              9,
              66
            ],
            "fieldUsedForNER": "transcription"
          }
        ],
        "preferredWikidataIds": [
          "http://www.wikidata.org/entity/Q64"
        ],
        "DBpediaIds": [
          "http://dbpedia.org/resource/Berlin",
          "http://de.dbpedia.org/resource/Berlin"
        ],
        "wikidataIds": [
          "http://www.wikidata.org/entity/Q614184",
          "http://www.wikidata.org/entity/Q2345711",
```

Response Code

```
200
```

Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "2789",
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 10:45:10 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

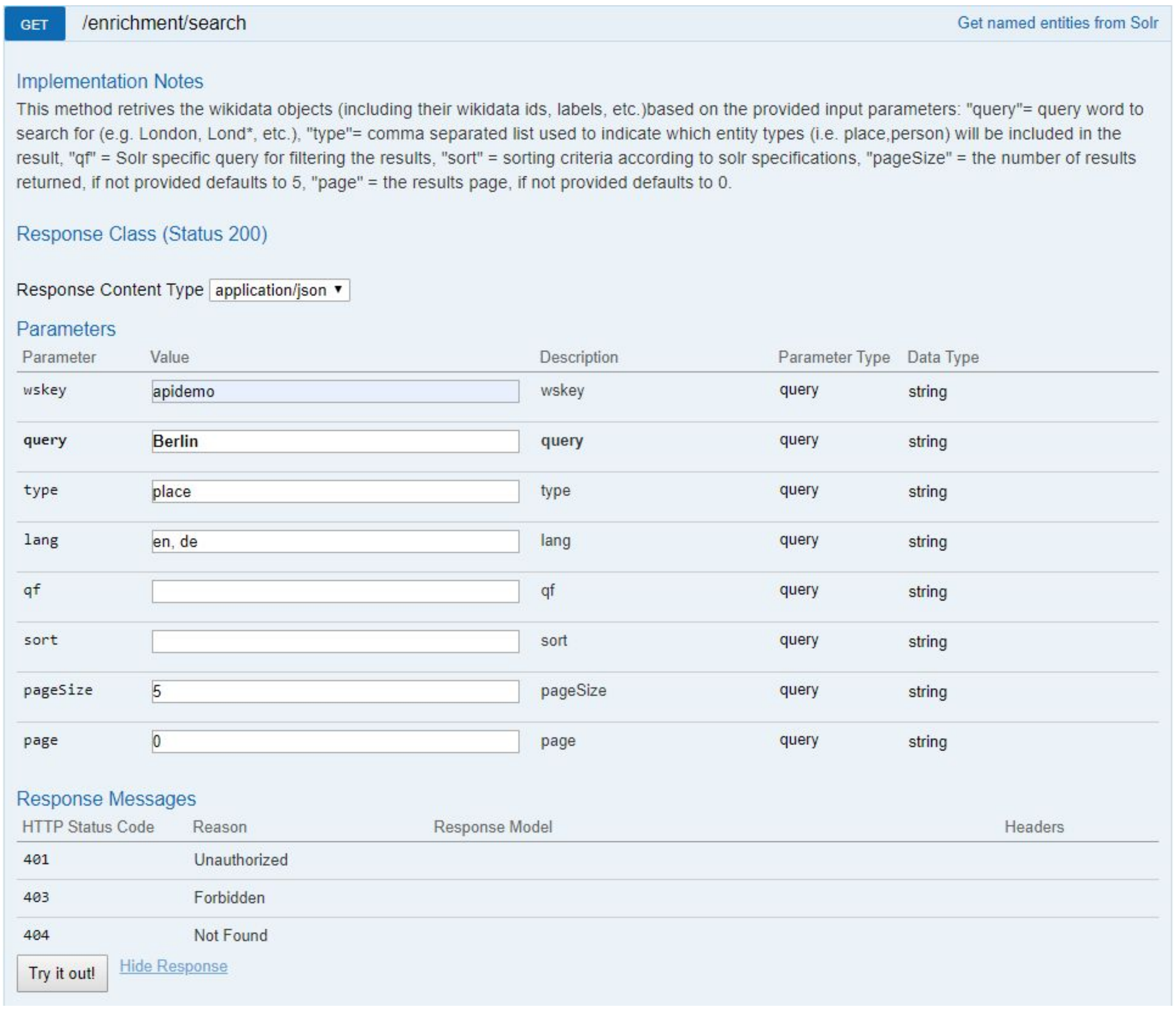
Figure 29: Retrieve Named Entities response for items

Entity Preview Services

The entity preview services retrieve the information about the found named entities in terms of the Wikidata related information. During the NER analysis, the Wikidata resources that are linked to the found entities in the analyzed texts are saved to the running Solr server. Solr is used for indexing and search of the Wikidata entities.

Search known entities

This service enables searching of Wikidata entities or resources based on the Solr query parameter (“query” parameter), the type of the entity (“type” parameter), the Solr specific filtering parameter (“qf” parameter), and the sorting criteria (“sort” parameter). Please refer to the Implementation Notes in Figure 30, for more details on the parameters descriptions.



GET /enrichment/search — Get named entities from Solr

Implementation Notes

This method retrives the wikidata objects (including their wikidata ids, labels, etc.)based on the provided input parameters: "query"= query word to search for (e.g. London, Lond*, etc.), "type"= comma separated list used to indicate which entity types (i.e. place,person) will be included in the result, "qf" = Solr specific query for filtering the results, "sort" = sorting criteria according to solr specifications, "pageSize" = the number of results returned, if not provided defaults to 5, "page" = the results page, if not provided defaults to 0.

Response Class (Status 200)

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
query	Berlin	query	query	string
type	place	type	query	string
lang	en, de	lang	query	string
qf		qf	query	string
sort		sort	query	string
pageSize	5	pageSize	query	string
page	0	page	query	string

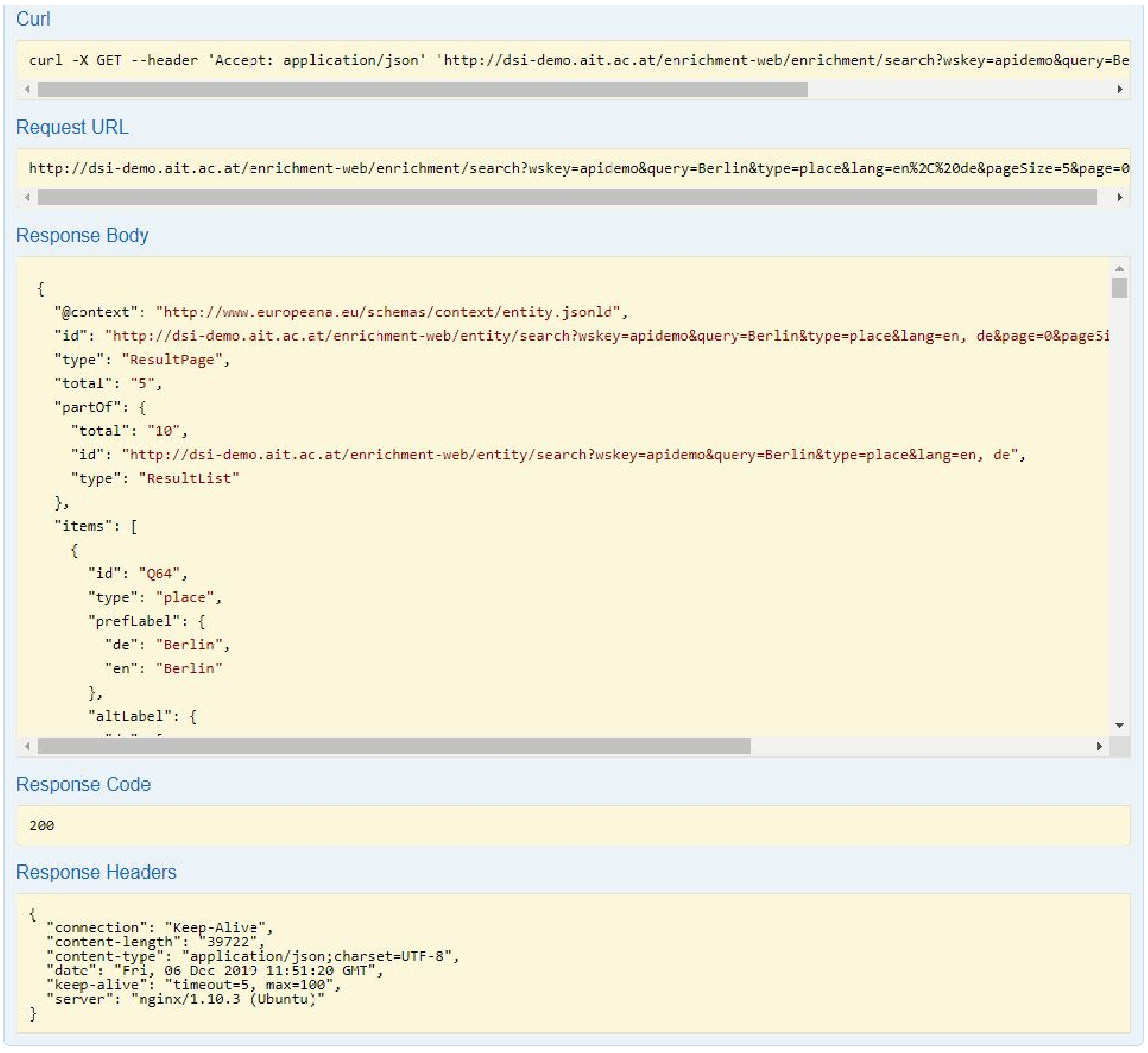
Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 30: Search known entities request

The response of the search method contains different fields that describe the given Wikidata resource is sketched in Figure 31 (i.e. prefLabel, altLabel, id, and type). For a more detailed description of all fields included in the entity preview please refer to https://www.wikidata.org/ and the mapping of those fields to the entity preview. The response is provided using the JSON format.



Curl

```
curl -X GET --header 'Accept: application/json' 'http://dsi-demo.ait.ac.at/enrichment-web/enrichment/search?wskey=apidemo&query=Be
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/enrichment/search?wskey=apidemo&query=Berlin&type=place&lang=en%2C%20de&pageSize=5&page=0
```

Response Body

```
{
  "@context": "http://www.europeana.eu/schemas/context/entity.jsonld",
  "id": "http://dsi-demo.ait.ac.at/enrichment-web/entity/search?wskey=apidemo&query=Berlin&type=place&lang=en, de&page=0&pageSi
  "type": "ResultPage",
  "total": "5",
  "partOf": {
    "total": "10",
    "id": "http://dsi-demo.ait.ac.at/enrichment-web/entity/search?wskey=apidemo&query=Berlin&type=place&lang=en, de",
    "type": "ResultList"
  },
  "items": [
    {
      "id": "Q64",
      "type": "place",
      "prefLabel": {
        "de": "Berlin",
        "en": "Berlin"
      },
      "altLabel": {
```

Response Code

```
200
```

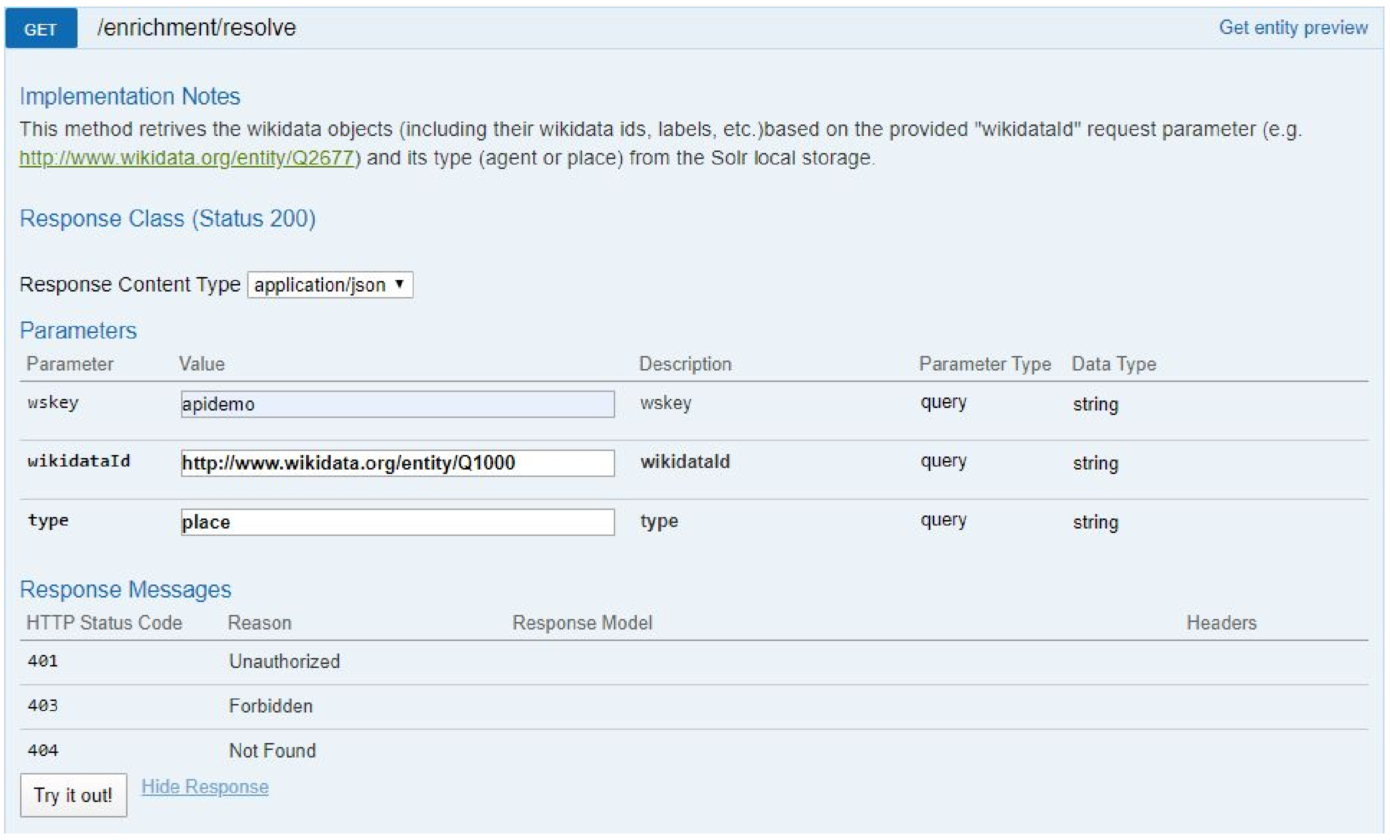
Response Headers

```
{
  "connection": "Keep-Alive",
  "content-length": "39722",
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 11:51:20 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 31: Response of the entity search service

Retrieve entity preview

This retrieve entity preview service returns the information about known Wikidata resources. While in the previous search method that uses Solr queries the user can specify several parameters to filter the results (e.g. language or Solr filtering parameter), here the search is based only on the provided Wikidata id and no additional filtering is provided. Therefore, for instance in Figure 33 for the response part we can see the "prefLabel" field containing the given entity in all languages.



GET /enrichment/resolve — Get entity preview

Implementation Notes

This method retrives the wikidata objects (including their wikidata ids, labels, etc.)based on the provided "wikidataId" request parameter (e.g. http://www.wikidata.org/entity/Q2677) and its type (agent or place) from the Solr local storage.

Response Class (Status 200)

Response Content Type: application/json

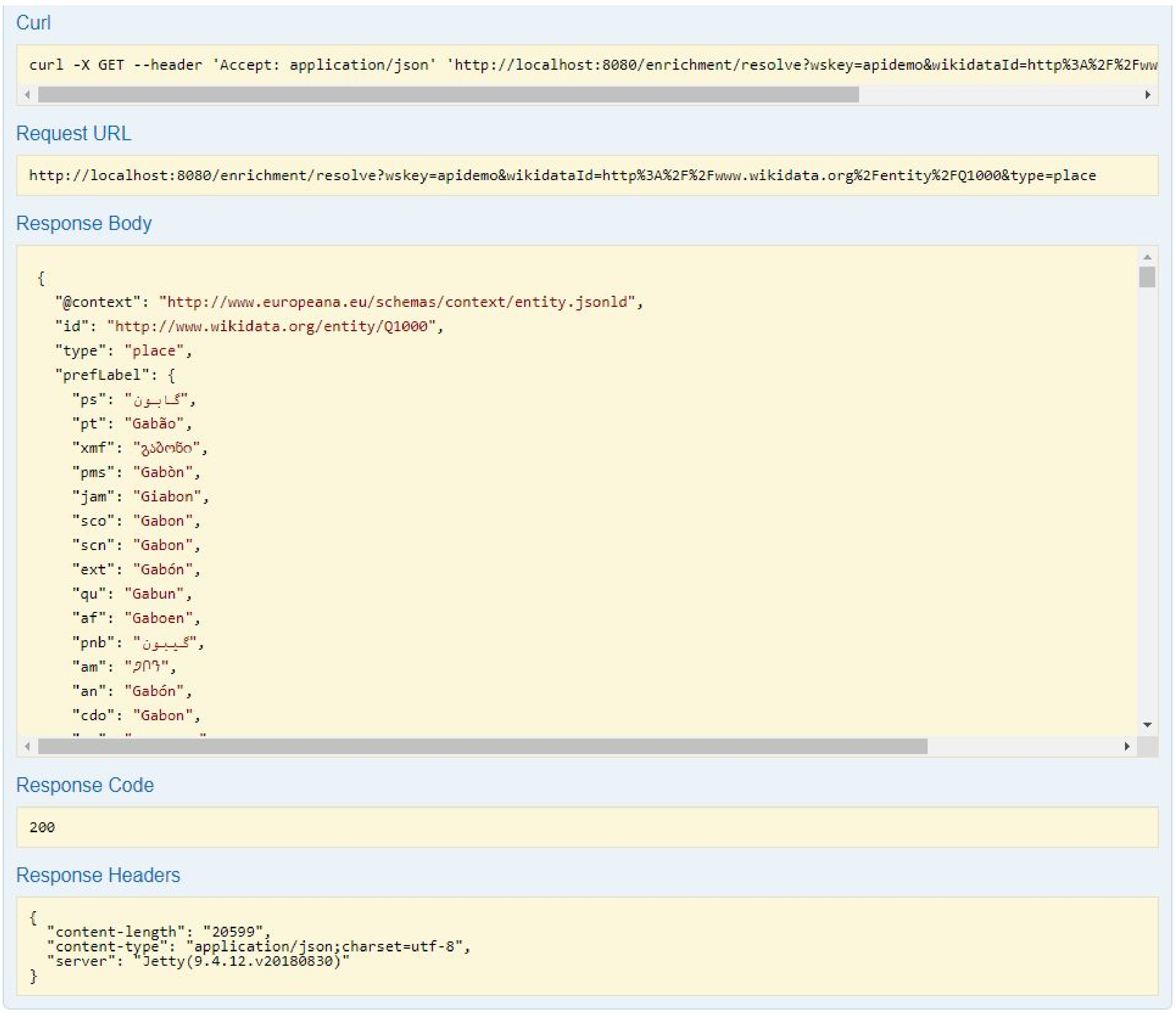
Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
wikidataId	http://www.wikidata.org/entity/Q1000	wikidataId	query	string
type	place	type	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 32: Retrieve entity preview



Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/enrichment/resolve?wskey=apidemo&wikidataId=http%3A%2F%2Fww
```

Request URL

```
http://localhost:8080/enrichment/resolve?wskey=apidemo&wikidataId=http%3A%2F%2Fwww.wikidata.org%2Fentity%2FQ1000&type=place
```

Response Body

```
{
  "@context": "http://www.europeana.eu/schemas/context/entity.jsonld",
  "id": "http://www.wikidata.org/entity/Q1000",
  "type": "place",
  "prefLabel": {
    "ps": "ګابون",
    "pt": "Gabão",
    "xmf": "გაბონი",
    "pms": "Gabòn",
    "jam": "Giabon",
    "sco": "Gabon",
    "scn": "Gabon",
    "ext": "Gabón",
    "qu": "Gabun",
    "af": "Gaboen",
    "pnb": "گیبون",
    "am": "ጋቦን",
    "an": "Gabón",
    "cdo": "Gabon",
```

Response Code

```
200
```

Response Headers

```
{
  "content-length": "20599",
  "content-type": "application/json;charset=utf-8",
  "server": "Jetty(9.4.12.v20180830)"
}
```

Figure 33: Entity preview response

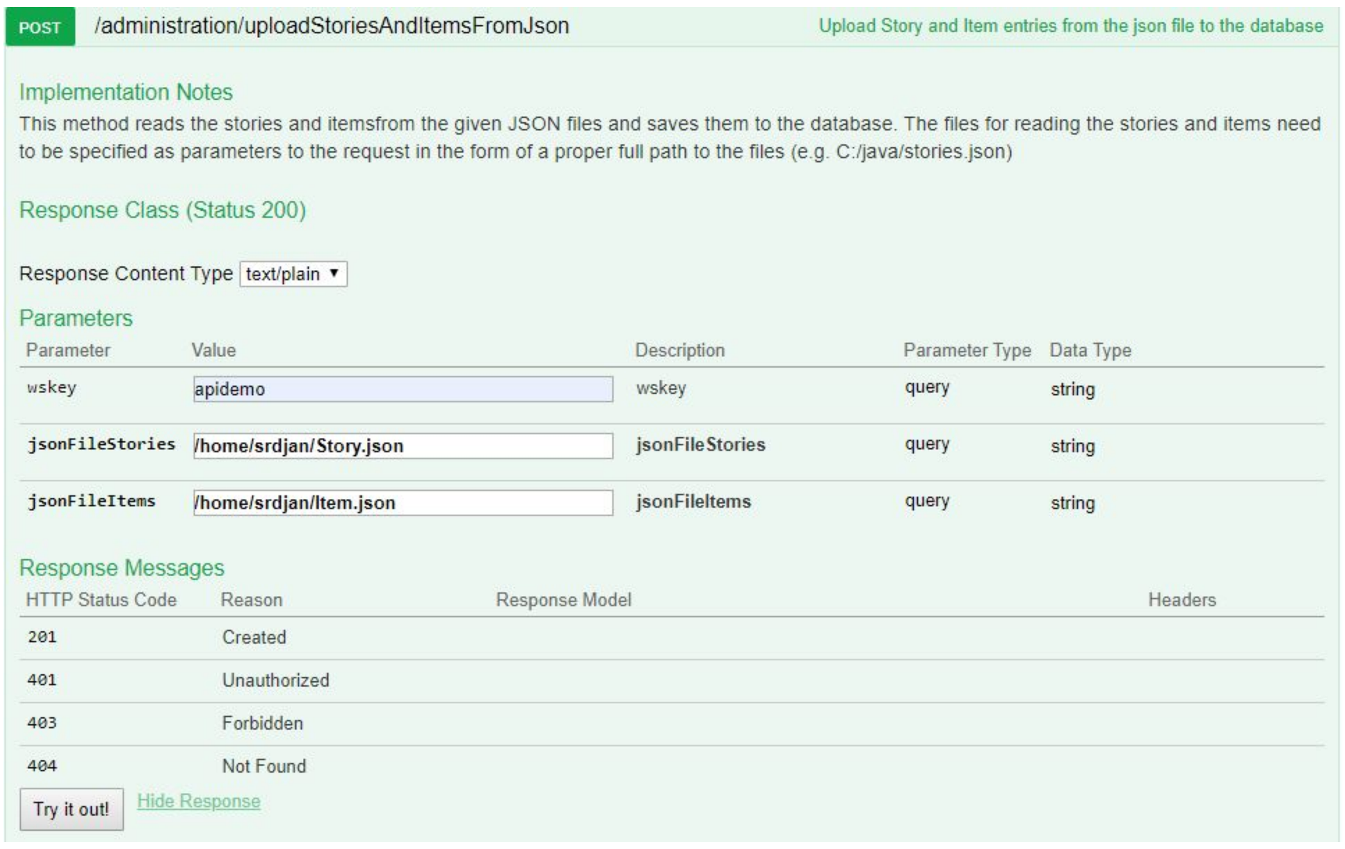
Co-financed by the Connecting Europe Facility of the European Union

Administration services

The administration services are used to import Transcribathon stories, items or their translations into the enrichment database. Stories and items can be imported both from a JSON file and individually through the body of the requests. The translations can be uploaded from the body of the requests. These services are intended to be used at the beginning for importing the data to the database in order to be analyses by any other services provided above.

Administration POST service for importing stories and items from JSON

This service aims at providing the possibility of uploading stories and items to the database from the JSON file. That file should contain a JSON array of stories and/or items to be uploaded. The Mongo Database interface is created and used to populate the database with the data from stories, items or their translations. In case an invalid file is provided an HTTP Bad Request exception is thrown with the corresponding message.


POST /administration/uploadStoriesAndItemsFromJson Upload Story and Item entries from the json file to the database

Implementation Notes

This method reads the stories and itemsfrom the given JSON files and saves them to the database. The files for reading the stories and items need to be specified as parameters to the request in the form of a proper full path to the files (e.g. C:/java/stories.json)

Response Class (Status 200)

Response Content Type text/plain

Parameters

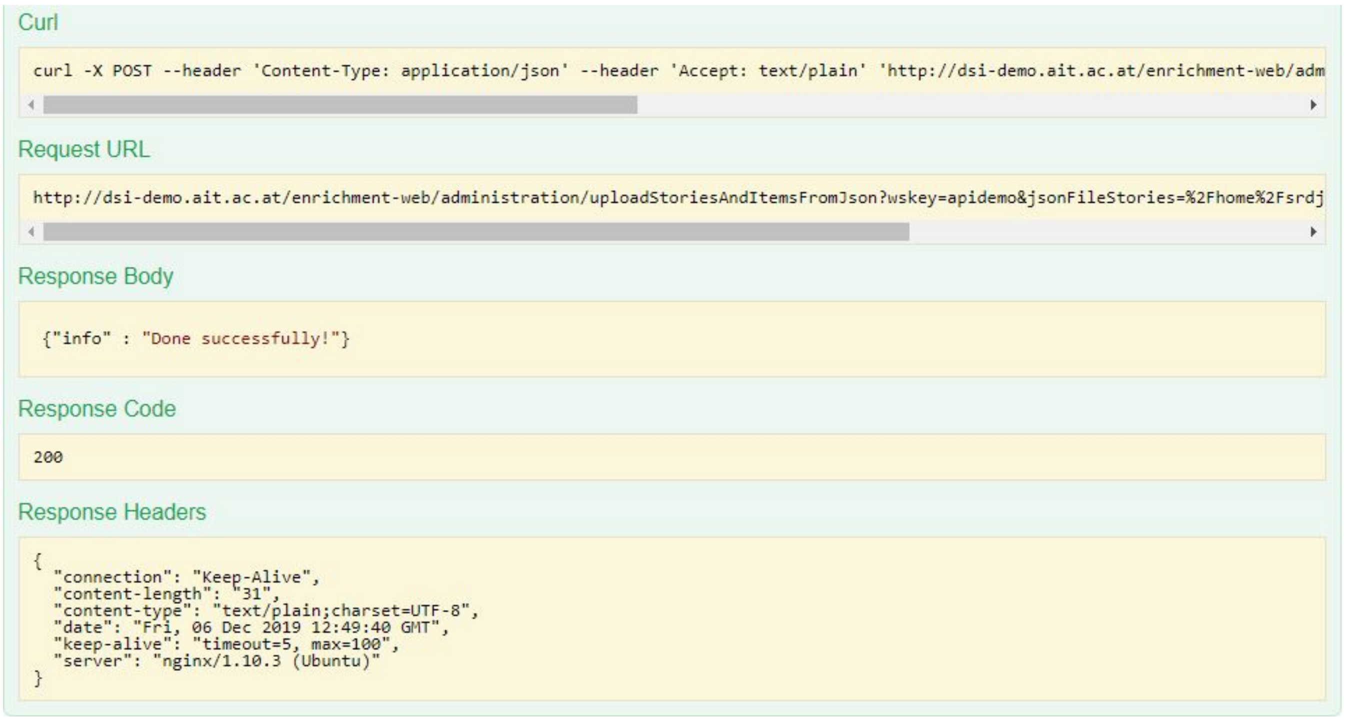
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
jsonFileStories	/home/srdjan/Story.json	jsonFileStories	query	string
jsonFileItems	/home/srdjan/Item.json	jsonFileItems	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Fig 34: REST endpoint for importing stories and items from JSON files


Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' 'http://dsi-demo.ait.ac.at/enrichment-web/adm
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/administration/uploadStoriesAndItemsFromJson?wskey=apidemo&jsonFileStories=%2Fhome%2Fsrdj
```

Response Body

```
{"info" : "Done successfully!"}
```

Response Code

```
200
```

Response Headers

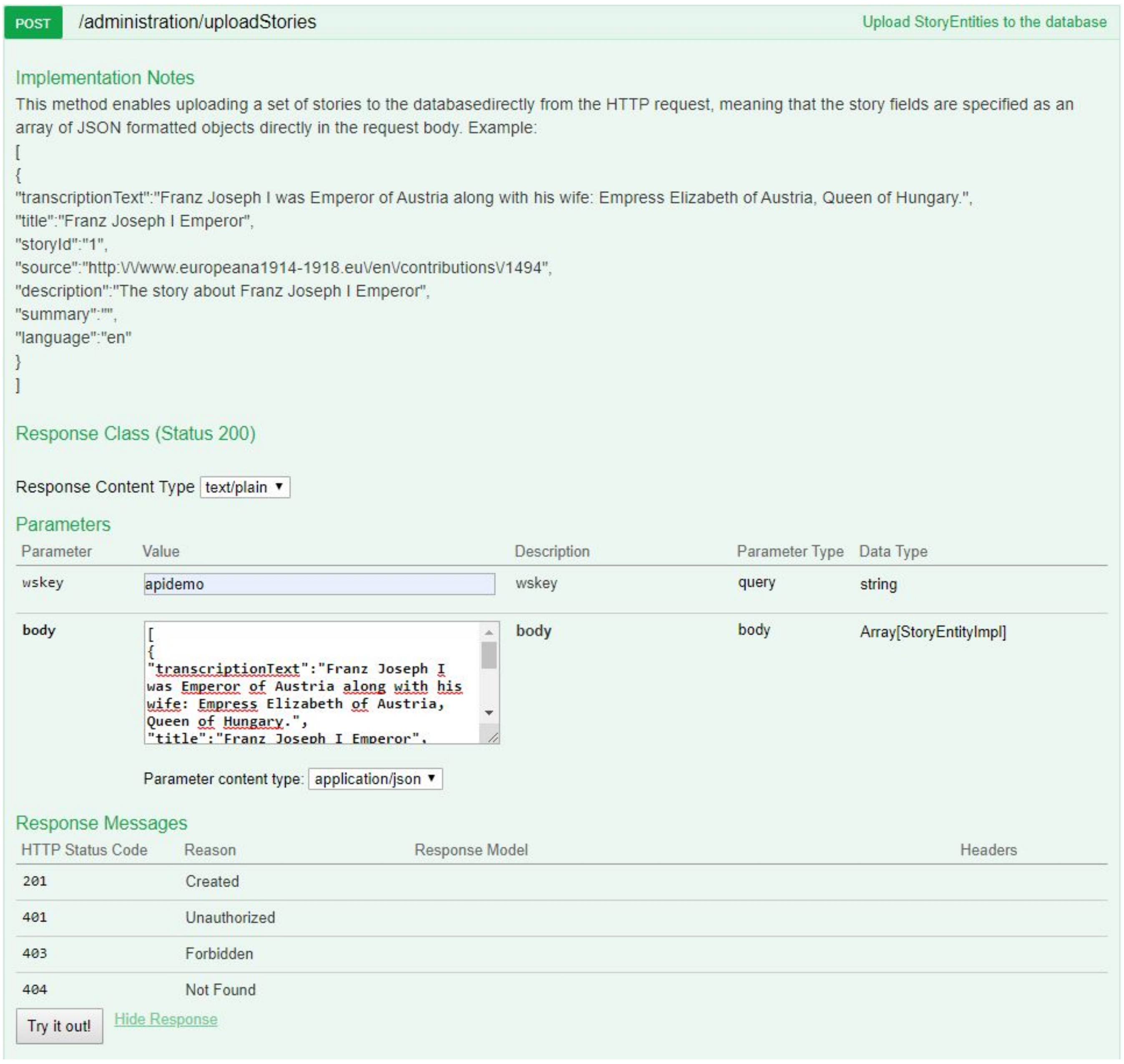
```
{
  "connection": "Keep-Alive",
  "content-length": "31",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 12:49:40 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Figure 35: Response of a successful import of stories and items from JSON files

Co-financed by the Connecting Europe Facility of the European Union

Administration service for importing stories over the request body

The request body data format in this service for directly uploading stories is a list of JSON objects with the story properties such as the storyId, title, source, description, summary, language and transcriptionText. A hash value for the transcriptionText is generated to enable easy verification of the transcriptionText updates. The Implementation Notes part contains an example of the request body that can be used to upload stories directly to the database. In case that some parameters are not provided correctly an HTTP Bad Request exception with the corresponding message is shown.


POST /administration/uploadStories — Upload StoryEntities to the database

Implementation Notes

This method enables uploading a set of stories to the databasedirectly from the HTTP request, meaning that the story fields are specified as an array of JSON formatted objects directly in the request body. Example:

```
[
{
"transcriptionText":"Franz Joseph I was Emperor of Austria along with his wife: Empress Elizabeth of Austria, Queen of Hungary.",
"title":"Franz Joseph I Emperor",
"storyId":"1",
"source":"http:\/\/www.europeana1914-1918.eu\/en\/contributions\/1494",
"description":"The story about Franz Joseph I Emperor",
"summary":"",
"language":"en"
}
]
```

Response Class (Status 200)

Response Content Type: text/plain

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
body	[{ "transcriptionText":"Franz Joseph I was Emperor of Austria along with his wife: Empress Elizabeth of Austria, Queen of Hungary.", "title":"Franz Joseph I Emperor",	body	body	Array[StoryEntityImpl]

Parameter content type: application/json

Response Messages

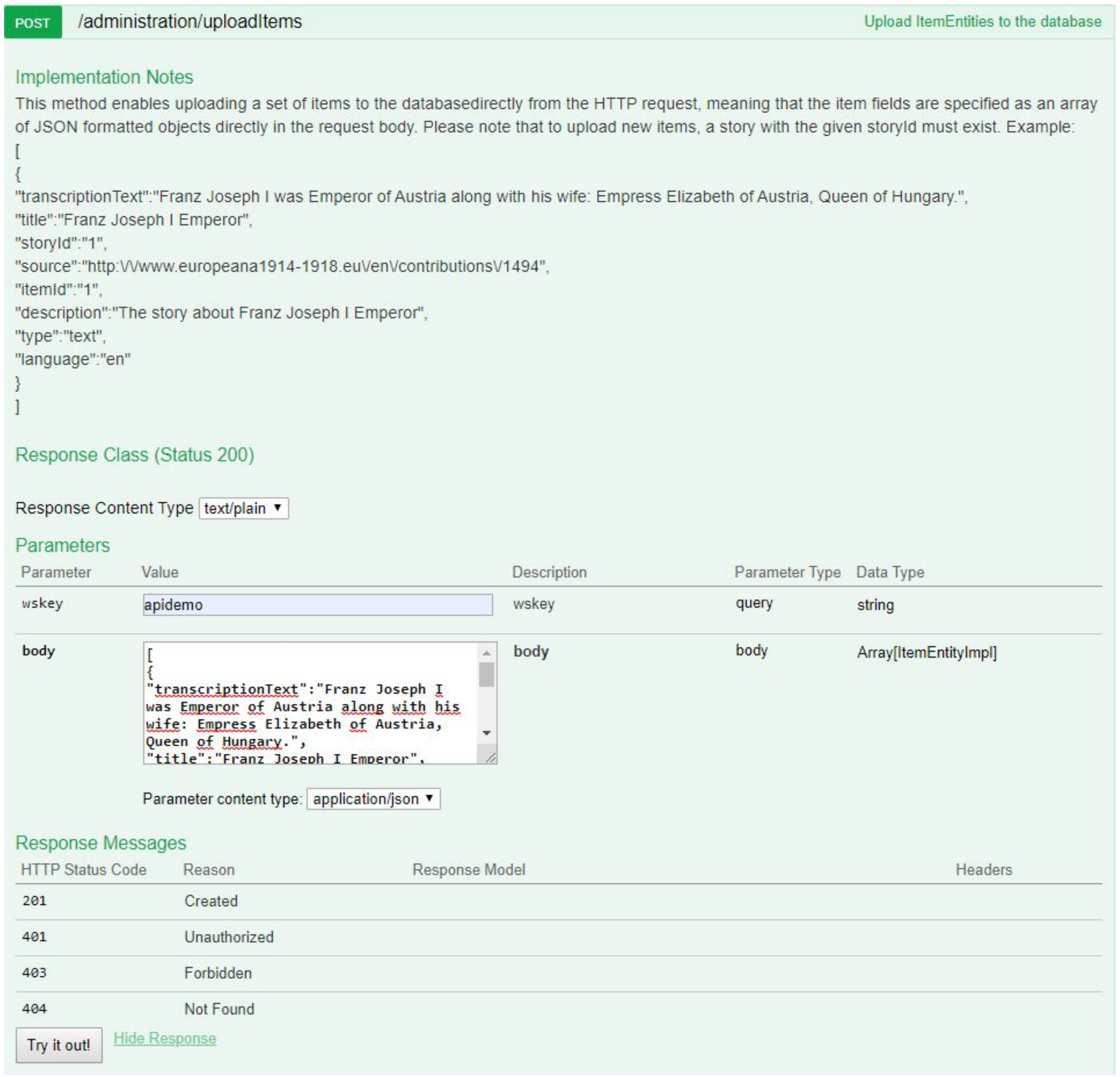
HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 36: REST endpoint for importing stories and items submitted through the request body

Co-financed by the Connecting Europe Facility of the European Union

Administration service for uploading items over the request body

This service uploads items directly from the request body to the enrichment database. The request body contains a list of JSON objects with the items properties like storyId, source, itemId, description, title, type (e.g. "letter" or "diary" or "picture"), language and transcriptionText. The same as for stories, a hash value for the transcriptionText is generated to enable easy verification of the transcriptionText updates. The error handling is the same as in case of uploading stories.



POST /administration/uploadItems — Upload ItemEntities to the database

Implementation Notes

This method enables uploading a set of items to the databasedirectly from the HTTP request, meaning that the item fields are specified as an array of JSON formatted objects directly in the request body. Please note that to upload new items, a story with the given storyId must exist. Example:

```
[
{
"transcriptionText":"Franz Joseph I was Emperor of Austria along with his wife: Empress Elizabeth of Austria, Queen of Hungary.",
"title":"Franz Joseph I Emperor",
"storyId":"1",
"source":"http:\/\/www.europeana1914-1918.eu\/en\/contributions\/1494",
"itemId":"1",
"description":"The story about Franz Joseph I Emperor",
"type":"text",
"language":"en"
}
]
```

Response Class (Status 200)

Response Content Type: text/plain

Parameters

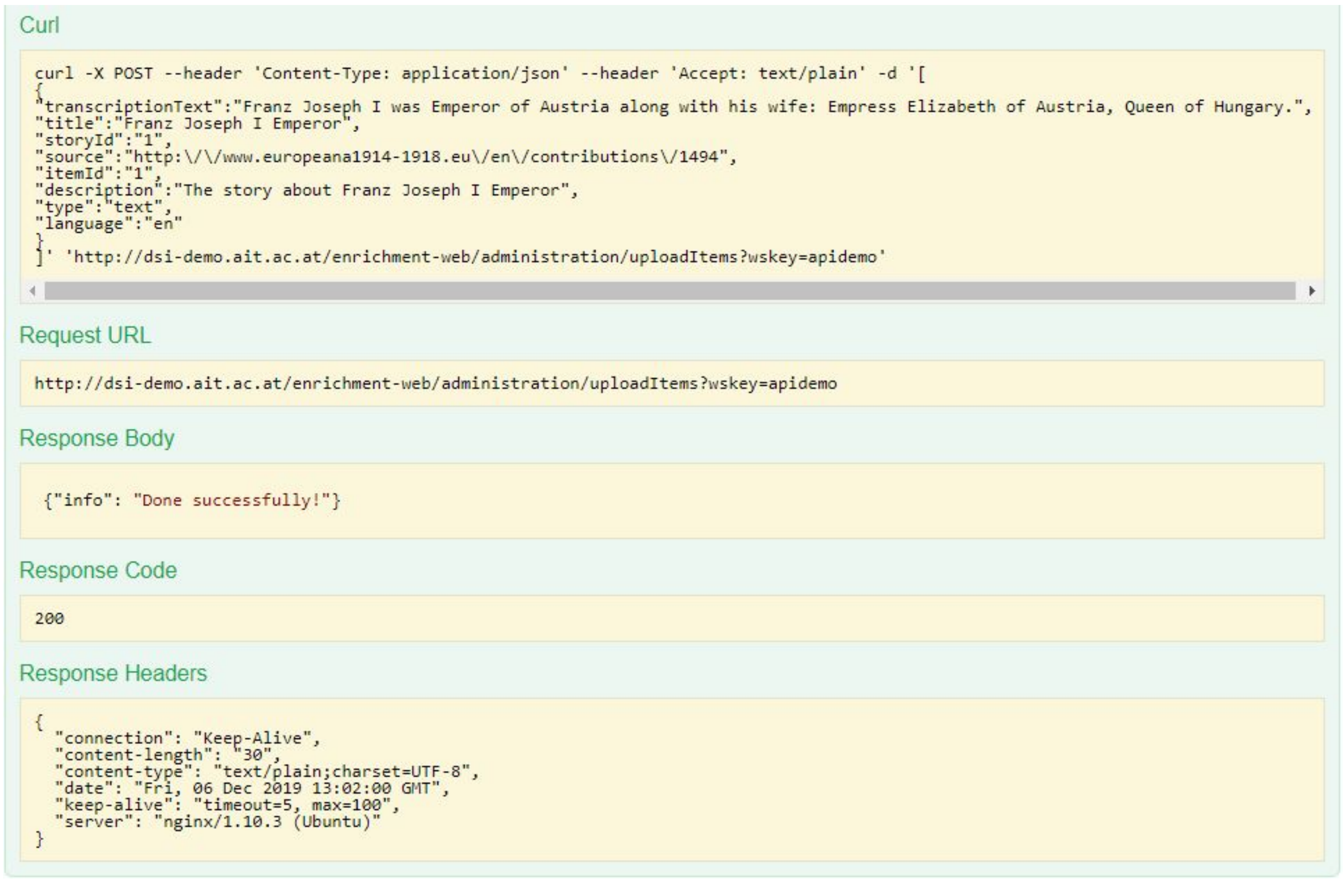
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
body	[{ "transcriptionText":"Franz Joseph I was Emperor of Austria along with his wife: Empress Elizabeth of Austria, Queen of Hungary.", "title":"Franz Joseph I Emperor",	body	body	Array[ItemEntityImpl]

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 37: REST endpoint for importing items submitted through the request body



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' -d '[
{
"transcriptionText":"Franz Joseph I was Emperor of Austria along with his wife: Empress Elizabeth of Austria, Queen of Hungary.",
"title":"Franz Joseph I Emperor",
"storyId":"1",
"source":"http:\/\/www.europeana1914-1918.eu\/en\/contributions\/1494",
"itemId":"1",
"description":"The story about Franz Joseph I Emperor",
"type":"text",
"language":"en"
}
]' 'http://dsi-demo.ait.ac.at/enrichment-web/administration/uploadItems?wskey=apidemo'
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/administration/uploadItems?wskey=apidemo
```

Response Body

```
{"info": "Done successfully!"}
```

Response Code

```
200
```

Response Headers

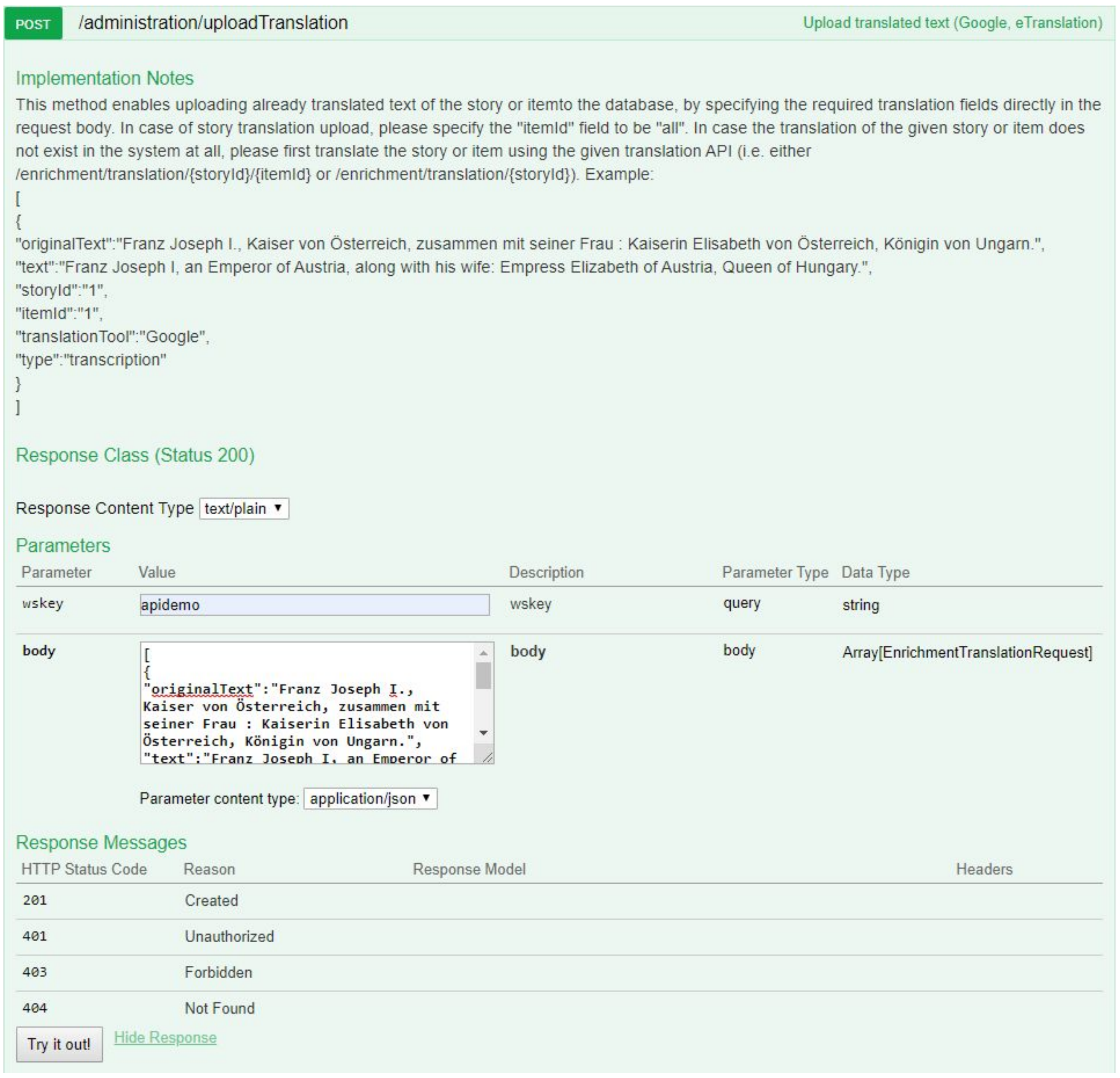
```
{
  "connection": "Keep-Alive",
  "content-length": "30",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 13:02:00 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Fig 38: Response of successful items import

Co-financed by the Connecting Europe
Facility of the European Union

Administration POST service for uploading the translations of items or stories

This service enables the upload of story or item translations to the database directly from the request body. Please refer to the Implementation Notes part in Figure 42 for the description of the query parameters. As in the case of the previous two administration services for uploading stories and items, in case of incorrectly provided parameters in the request body an HTTP exception with the corresponding message is shown.



POST /administration/uploadTranslation — Upload translated text (Google, eTranslation)

Implementation Notes

This method enables uploading already translated text of the story or itemto the database, by specifying the required translation fields directly in the request body. In case of story translation upload, please specify the "itemId" field to be "all". In case the translation of the given story or item does not exist in the system at all, please first translate the story or item using the given translation API (i.e. either /enrichment/translation/{storyId}/{itemId} or /enrichment/translation/{storyId}). Example:

```
[
{
"originalText":"Franz Joseph I., Kaiser von Österreich, zusammen mit seiner Frau : Kaiserin Elisabeth von Österreich, Königin von Ungarn.",
"text":"Franz Joseph I, an Emperor of Austria, along with his wife: Empress Elizabeth of Austria, Queen of Hungary.",
"storyId":"1",
"itemId":"1",
"translationTool":"Google",
"type":"transcription"
}
]
```

Response Class (Status 200)

Response Content Type: text/plain

Parameters

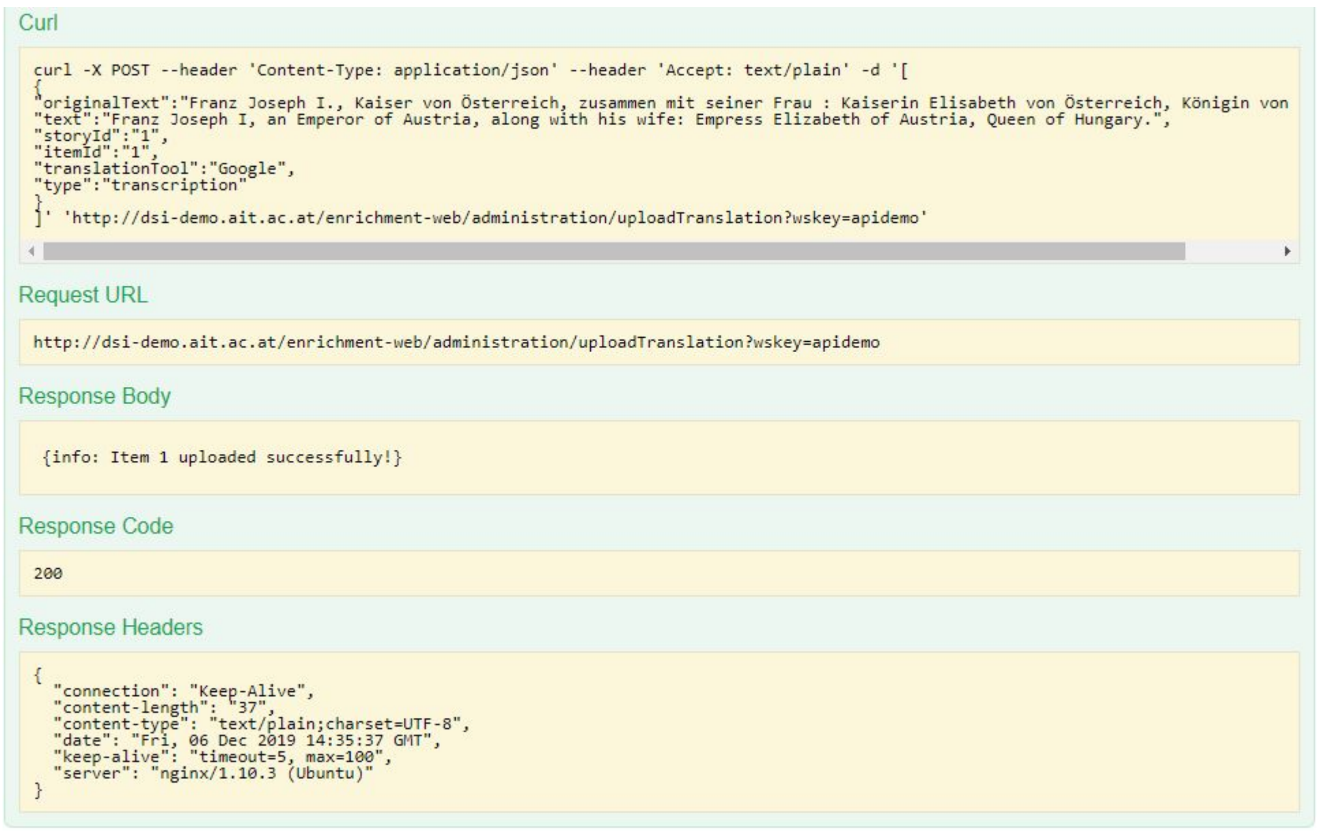
Parameter	Value	Description	Parameter Type	Data Type
wskey	apidemo	wskey	query	string
body	[{ "originalText":"Franz Joseph I., Kaiser von Österreich, zusammen mit seiner Frau : Kaiserin Elisabeth von Österreich, Königin von Ungarn.", "text":"Franz Joseph I, an Emperor of	body	body	Array[EnrichmentTranslationRequest]

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Fig 39: REST service for importing translations of items or stories



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain' -d '[
{
"originalText":"Franz Joseph I., Kaiser von Österreich, zusammen mit seiner Frau : Kaiserin Elisabeth von Österreich, Königin von
"text":"Franz Joseph I, an Emperor of Austria, along with his wife: Empress Elizabeth of Austria, Queen of Hungary.",
"storyId":"1",
"itemId":"1",
"translationTool":"Google",
"type":"transcription"
}
]' 'http://dsi-demo.ait.ac.at/enrichment-web/administration/uploadTranslation?wskey=apidemo'
```

Request URL

```
http://dsi-demo.ait.ac.at/enrichment-web/administration/uploadTranslation?wskey=apidemo
```

Response Body

```
{info: Item 1 uploaded successfully!}
```

Response Code

```
200
```

Response Headers

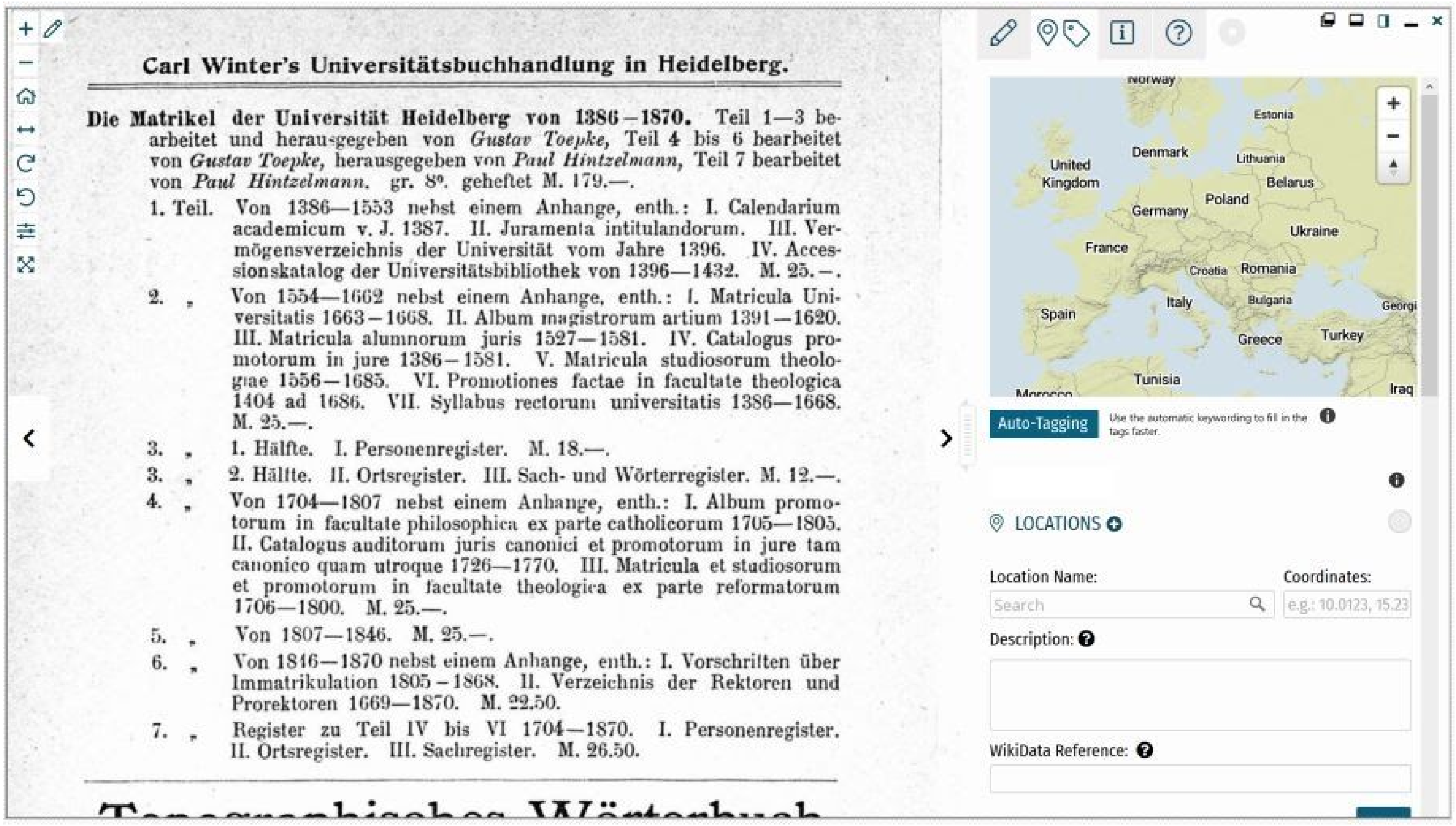
```
{
  "connection": "Keep-Alive",
  "content-length": "37",
  "content-type": "text/plain;charset=UTF-8",
  "date": "Fri, 06 Dec 2019 14:35:37 GMT",
  "keep-alive": "timeout=5, max=100",
  "server": "nginx/1.10.3 (Ubuntu)"
}
```

Fig 40: Response for successful import of translations of items or stories

Co-financed by the Connecting Europe
Facility of the European Union

Integration of Enrichment Services in Transcribathon Tool

The semantic enrichment services are used to support end users when enhancing descriptions and providing contextual information related to the transcribed documents.The Auto-Tagging functionality is used to discover named entities referenced within the transcription text and to map them to existing entries from linked data repositories. This functionality is available to be used by the users once the transcription is complete, see Figure 41.



Carl Winter's Universitätsbuchhandlung in Heidelberg.

Die Matrikel der Universität Heidelberg von 1386–1870. Teil 1—3 bearbeitet und herausgegeben von *Gustav Toepke*, Teil 4 bis 6 bearbeitet von *Gustav Toepke*, herausgegeben von *Paul Hintzelmann*, Teil 7 bearbeitet von *Paul Hintzelmann*. gr. 8⁰. geheftet M. 179.—.

1. Teil. Von 1386—1553 nebst einem Anhange, enth.: I. Calendarium academicum v. J. 1387. II. Juramenta intitulandorum. III. Vermögensverzeichnis der Universität vom Jahre 1396. IV. Accessionskatalog der Universitätsbibliothek von 1396—1432. M. 25.—.

2. „ Von 1554—1662 nebst einem Anhange, enth.: I. Matricula Universitatis 1663—1668. II. Album magistrorum artium 1391—1620. III. Matricula alumnorum juris 1527—1581. IV. Catalogus promotorum in jure 1386—1581. V. Matricula studiosorum theologiae 1556—1685. VI. Promotiones factae in facultate theologica 1404 ad 1686. VII. Syllabus rectorum universitatis 1386—1668. M. 25.—.

3. „ 1. Hälfte. I. Personenregister. M. 18.—.

3. „ 2. Hälfte. II. Ortsregister. III. Sach- und Wörterregister. M. 12.—.

4. „ Von 1704—1807 nebst einem Anhange, enth.: I. Album promotorum in facultate philosophica ex parte catholicorum 1705—1805. II. Catalogus auditorum juris canonici et promotorum in jure tam canonico quam utroque 1726—1770. III. Matricula et studiosorum et promotorum in facultate theologica ex parte reformatorum 1706—1800. M. 25.—.

5. „ Von 1807—1846. M. 25.—.

6. „ Von 1846—1870 nebst einem Anhange, enth.: I. Vorschriften über Immatrikulation 1805–1868. II. Verzeichnis der Rektoren und Prorektoren 1669—1870. M. 22.50.

7. „ Register zu Teil IV bis VI 1704—1870. I. Personenregister. II. Ortsregister. III. Sachregister. M. 26.50.

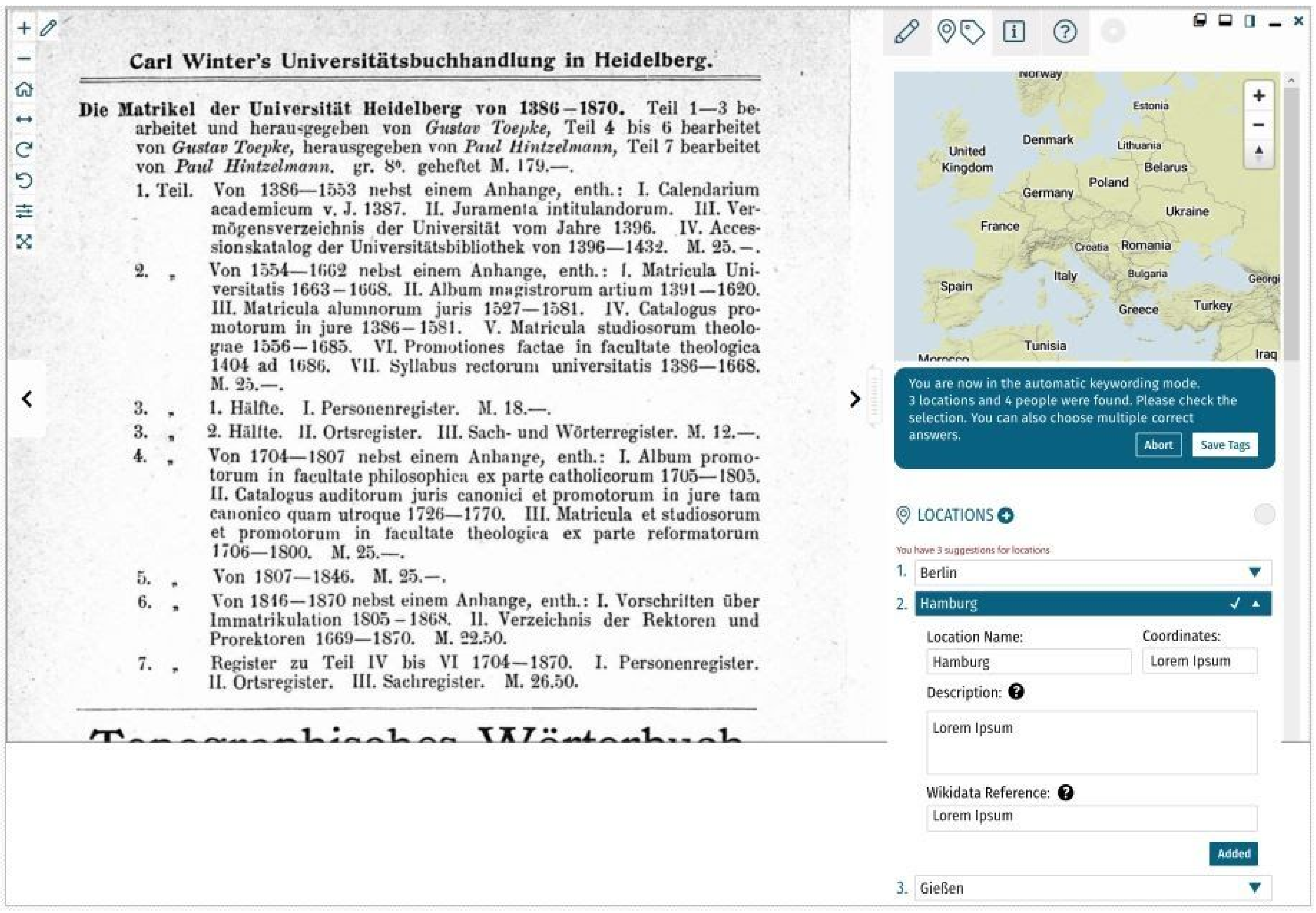
Auto-Tagging
Use the automatic keywording to fill in the tags faster.

LOCATIONS

Location Name: Search
Coordinates: e.g.: 10.0123, 15.23
Description:
WikiData Reference:

Fig 41: Invocation of semantic enrichment services in Transcribathon Frontend

Previous experiments indicate that the detection and linking of places is performed with a good precision, around 80%. Detection of person names is also good, but many of the persons named in historical documents are not available and will probably not be available in linked data repositories. Consequently, in order to ensure the expected quality of semantic enrichments, they must be manually verified and validated by transcribathon users.



Carl Winter's Universitätsbuchhandlung in Heidelberg.

Die Matrikel der Universität Heidelberg von 1386–1870. Teil 1—3 bearbeitet und herausgegeben von *Gustav Toepke*, Teil 4 bis 6 bearbeitet von *Gustav Toepke*, herausgegeben von *Paul Hintzelmann*, Teil 7 bearbeitet von *Paul Hintzelmann*. gr. 8⁰. geheftet M. 179.—.

1. Teil. Von 1386—1553 nebst einem Anhange, enth.: I. Calendarium academicum v. J. 1387. II. Juramenta intitulandorum. III. Vermögensverzeichnis der Universität vom Jahre 1396. IV. Accessionskatalog der Universitätsbibliothek von 1396—1432. M. 25.—.

2. „ Von 1554—1662 nebst einem Anhange, enth.: I. Matricula Universitatis 1663—1668. II. Album magistrorum artium 1391—1620. III. Matricula alumnorum juris 1527—1581. IV. Catalogus promotorum in jure 1386—1581. V. Matricula studiosorum theologiae 1556—1685. VI. Promotiones factae in facultate theologica 1404 ad 1686. VII. Syllabus rectorum universitatis 1386—1668. M. 25.—.

3. „ 1. Hälfte. I. Personenregister. M. 18.—.

3. „ 2. Hälfte. II. Ortsregister. III. Sach- und Wörterregister. M. 12.—.

4. „ Von 1704—1807 nebst einem Anhange, enth.: I. Album promotorum in facultate philosophica ex parte catholicorum 1705—1805. II. Catalogus auditorum juris canonici et promotorum in jure tam canonico quam utroque 1726—1770. III. Matricula et studiosorum et promotorum in facultate theologica ex parte reformatorum 1706—1800. M. 25.—.

5. „ Von 1807—1846. M. 25.—.

6. „ Von 1846—1870 nebst einem Anhange, enth.: I. Vorschriften über Immatrikulation 1805–1868. II. Verzeichnis der Rektoren und Prorektoren 1669—1870. M. 22.50.

7. „ Register zu Teil IV bis VI 1704—1870. I. Personenregister. II. Ortsregister. III. Sachregister. M. 26.50.

You are now in the automatic keywording mode. 3 locations and 4 people were found. Please check the selection. You can also choose multiple correct answers.
Abort | Save Tags

LOCATIONS

You have 3 suggestions for locations

1. Berlin
2. Hamburg

Location Name: Hamburg
Coordinates: Lorem Ipsum
Description: Lorem Ipsum
Wikidata Reference: Lorem Ipsum
Added

3. Gießen

Fig 42: Verification and approval of semantic enrichments by end users

Figure 42 shows the list of discovered places displayed for approval by users. The approved enrichments are saved within the database of Transcribathon platform.

Conclusions and further Development

This document presents the technical documentation for the final version of the semantic enrichment and quality control services. All available services are presented and each has been documented with an explanatory example. The modifications and improvements against the previous version of the services are described. The benefits of these services for the materials available through the Transcribathon platform include:

- improving the accessibility of historical documents for the large public by machine translation where the documents can be translated to the language a user is familiar with
- improving the understandability of historical documents by integrating relevant information from Linked Open Data resources (i.e. Europeana Entity Collection, Wikidata, DbPedia)
- improving the discoverability of materials in Transcribathon platform by enabling search in different languages, interlinking trough semantic enrichments or enhanced browsing capabilities (e.g. based on newly identified locations, persons or geo-locations)

Future work will focus on experimentation of automatic topic detection and recommendation of keywords used for tagging and categorization of historical documents.

References

Europeana Annotation API: https://pro.europeana.eu/resources/apis/annotations

Europeana Entity API: https://pro.europeana.eu/resources/apis/entity

Web Annotation Data Model: https://www.w3.org/TR/annotation-model/

Annotation use cases submitted to Annotations Task Force:

https://docs.google.com/document/d/1af56Omq1GP1xLVvXHywxQXazWqEfhzRfMTbo6lwv5QU/edit

Specifications for Enrichment Services:
https://docs.google.com/document/d/1V5cK1Y1jp-Bn087FTN-aldOv3pbGqffUOxWV2Xp3ZJk/edit

Mapping entity previews for Wikidata entities:

https://docs.google.com/spreadsheets/d/1EakZVo0R7MXzIHGt_I5rW2IPi9Z7uj38DHmsfgQaUjw/edit?usp=drive_web&ouid=1145694
56110932326926

Annex 1: Tools used for the evaluation of Named Entity Recognition, classification and linking

Stanford NER (Version: 3.9.1)
Stanford NER was the first tool applied on the corpus, with two different standard English classifiers. One of these classifiers is based on a 4 class model (Location, Person, Organization and Misc) which were trained on the "CoNLL 2003 eng.train" dataset and the other one is based on a 3 class model (Location, Person and Organization) which also included the MUC 6 and MUC 7 training data set (for more information, see https://nlp.stanford.edu/software/CRF-NER.shtml). In addition to these two Stanford classifier models, the German model was also applied on Eduard Scheer's War diaries.

NLTK (Version: 3.3)
The Python-based named entity recognition tool NLTK (Natural Language Toolkit) was added as a comparison point for the Stanford NER classifiers. We used it with the default configuration and default classifier, which is also based on a 3 class model (Location, Person and Organization).

spaCy (Version: 2.0.16), Flair (Version: 0.4)
Flair and spaCy were applied on the corpus to serve as further reference to Stanford and NLTK. Both tools were used with the default settings and default classifiers.

DBpedia spotlight (Version: 0.7.1)
DBpedia spotlight provides an integrated solution for named entity recognition, classification and linking. Initially, the Candidates web interface of DBpedia spotlight was used to evaluate the tool performance on English text translations (i.e. using English version of DBpedia spotlight). Later on, the Docker based installation was used to retrieve named entities using English, German and Italian versions of DBpedia Spotlight.

Co-financed by the Connecting Europe
Facility of the European Union

TINT (Version: 0.2)
TINT NER is the first tool we used for performing named entity recognition for the document in Italian language. The NER functionality included in Tint is based on the CRF Classifier included in Stanford CoreNLP. The model provided with TINT is trained on the Italian Content Annotation Bank (I-CAB), containing around 180,000 words taken from the Italian newspaper L'Adige, and used for the Entity Recognition task at Evalita 2009, the Temporal Expression Recognition and Normalization Task at Evalita 2007 and the Named Entity Recognition Task at Evalita 2007.

Dandelion
Dandelion is a named entity extraction & linking API that performs very well even on short texts, on which many other similar services do not. It currently works on texts in English, French, German, Italian, Portuguese, Russian, Spanish and many other languages. With this API it is possible to automatically tag texts, extracting Wikipedia entities and enriching data. Dandelion APIs were also tested for comparison.

Co-financed by the Connecting Europe Facility of the European Union