Deliverable 3.4

Web Services API

The project has received funding from the European Community (ICT-PSP 4th call)
under Grant Agreement n° 270915.

DELIVERABLE

Project Acronym:	**Bologna**
Grant Agreement number:	**270915**
Project Title:	**Bologna Translation Service**

Deliverable 3.4

Web Services API

Revision: 4

Authors:

Kim Scholte, CL

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	✓
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Rev.	Date	Author	Organisation	Description
1	26 Jan 2012	Kim Scholte	CL	Initial version
2	2 Feb 2012	Kim Scholte	CL	Review
3	6 Feb 2012	Heidi Depraetere	CL	Text edits
4	8 Feb 2012	Joeri Van de Walle	CL	Final review

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise.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.

Executive Summary

This deliverable provides a technical overview of the web services currently available with the Bologna Translation Service. These web services can be used by users to implement the Bologna Translation Service into their back-end systems.

Table of Contents

1. Introduction

This deliverable gives an overview of the web services currently available with the Bologna Translation Service.

Web services are only available for academic users and allow integration of the Bologna Translation Service into universities' or departments' back-end softwares.

2. RESTful Web Service

This topic describes the RESTful (REpresentational State Transfer) web service in its current status. Two additional deliverables are scheduled at month 18 and 24 which may include updates to the web service.

2.1 Introduction to REST

RESTful web services make use of the hypertext transfer protocol (http) to manage the creation and manipulation of objects. For the web service of the Bologna Translation Service these objects will usually be documents.

In HTTP we have 4 different actions that try to simulate what are known as CRUD (Create, Read, Update and Delete) actions when working with relational databases. These CRUD actions are all we need to submit a document and get back the translated version.

HTTP Action	CRUD	Meaning in REST
GET	Read	Download or retrieve a certain object or collection of objects. E.g. `GET /translation/5` or `GET /translation/5/status`
POST	Create	Create a new object. E.g. `POST /translation`
PUT	Update	Update a specified object or a collection of objects. E.g. `PUT /translation/5`
DELETE	Delete	Delete the given object or collection of objects. E.g. `DELETE /translation/5`

REST actions usually return a JSON[1] object when dealing with complex objects or a data stream when dealing with binary objects. Most programming languages have libraries in place to handle JSON objects, so they can be easily integrated into the application.

[1] http://en.wikipedia.org/wiki/JSON

2.2 Introduction to BTS REST

The Bologna Translation Service REST web service provides two options for translating documents:

- **Asynchronous:** The Bologna Translation Service allows users to make asynchronous requests. This means that when the document is uploaded a reference ID, called the asynchronous ID, will be returned and the connection to the server will be closed immediately. The ID can then be used to check the status of the request. When a request is reported as finished, the ID can be used to download the request. The asynchronous method is the recommended option for requesting translations.

- **Synchronous:** Small documents that need no post-editing can be sent in a synchronous way. This means that the document is sent to the server, and a connection is kept open until the translated document is returned. Most applications, however, impose a limit on how long a connection can be kept inactive. In case the translation is not finished before this limit the connection is closed, and the translation will be lost. When post-editing is required, this limit will certainly be reached. Therefore, synchronous translations should only be used for small documents that do not require post-editing.

It is important to understand the following concepts when using the REST web services of the Bologna Translation Service:

- **Username:** The username of the user who initiates the web service request. This is the same as the username used to log in to the BTS Portal. This will typically be the e-mail address of the user.

- **Password:** The password as set by the user in the BTS Portal. Note that this is not the same password as the one used to log in to the BTS Portal. It is the password set by the user on the web service configuration panel in the portal.

- **RequestTime:** The request time is the time when the request was created. It will be used to build the *secret* (see later), but will also be used as a counter measure against replay attacks. Only a limited difference between this time and the time on the server will be allowed, so in case the secret is stolen, it's only usable for a limited amount of time. Beside this security measure, an SSL connection is also used to encrypt the traffic.

- **Secret:** The secret is built using the request time, the username, and the password and will be used by the Bologna Translation Service to authenticate the user. A secret is used instead of a (hashed) password so the user never has to send the password itself over the wire. A description on how to create the secret follows in a later chapter.

- **AsyncID:** The asynchronous ID will be used to uniquely identify a translation request. It will be returned when sending the document to BTS, and can be used to check the status of the request or download the final translation.

- **Target and Source Languages:** These keywords represent the languages to use for the request. ISO 639.2 3-letter codes must be used to specify languages. For the 21 languages that have alternative codes for bibliographic or terminology purposes, use the bibliographic alternative ("B").

- **FileType:** This represents the file type of the document that is sent for translation. Possible options are: TXT, HTML, XML, DOCX, PPTX, XLSX, CSV.

- **Host:** The host name of the server. Will be published when the service is made publicly available.

In the sections that follow these keywords will be enclosed in curly brackets ("{…}") whenever they are used.

2.3 Creating asynchronous requests

Creating an asynchronous translation request can be done by `POST`-ing the document to the following URL:

```
https://{HOST}/translation/{username}/{secret}/{requestTime}/{fileType}/{srcLang}/{trgLang}/{domain}
```

The data that is posted should be made up of a multi-part message where the document to translate is stored in a body-part called "content".

Posting to this URL will return a JSON object that contains the asynchronous id of the request:

```
{"asyncId":11,"message":"New document received.","status":"RECEIVED"}
```

Example:

```
POST /translation/kim@cl.com/hN...1U=/08-02-2012%2015:58:52%20CET/html/DUT/ENG/Legal HTTP/1.1
Content-Type: multipart/form-data; boundary=Boundary 1 2003560079 1328713133350
MIME-Version: 1.0
User-Agent: Java/1.6.0_22
Host: bts1.bologna-translation.eu:12344
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Transfer-Encoding: chunked

81
--Boundary 1 2003560079 1328713133350
Content-Type: application/octet-stream
Content-Disposition: form-data; name="content"


6c8d
<html>

<head><base
href="http://www.ehsal.net/hubects/ectsfiche.asp?ectsnr=14527&taal=N&mod=1107&stdj=2HZ&acj=2011" />

  <link rel="stylesheet" type="text/css" media="scre

......

  </table>

 </body>
</html>
--Boundary_1_2003560079_1328713133350--

0
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 08 Feb 2012 14:58:54 GMT

45
{"asyncId":11,"message":"New document received.","status":"RECEIVED"}
0
```

2.4 Query the status of a asynchronous request

To get the status of an asynchronous request, a `GET` has to be requested on the following URL:

```
https://{HOST}/translation/{username}/{secret}/{requestTime}/{asyncId}/status
```

This will return a JSON object that contains the status of the request. In case processing the request failed, the reason is included in the status message:

```
{"asyncId":4,"message":"Language pair not supported.","status":"FAILED"}
```

```
{"asyncId":4,"message":"Ready for download.","status":"FINISHED"}
```

Possible return statuses are:

RECEIVED	*CANCELLED*
TRANSLATING	*PROCESSING*
WAITING_FOR_POSTEDITOR	*FAILED*
FINISHED	*WAITING_FOR_APPROVAL*

Example:

```
GET /translation/kim@cl.com/KXqs..........U8=/08-02-2012%2017:30:12%20CET/11/status
HTTP/1.1
Content-Type: multipart/form-data
User-Agent: Java/1.6.0 22
Host: bts1.bologna-translation.eu:12344
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 08 Feb 2012 16:30:13 GMT

37
{"asyncId":11,"message":"FINISHED","status":"FINISHED"}
0
```

2.5 Downloading the translation of a asynchronous request

Downloading the translation of an uploaded document can be done in the same way as checking its status, i.e. by requesting a `GET`, but needs a different URL:

```
https://{HOST}/translation/{username}/{secret}/{requestTime}/{asyncId}
```

This will return the translated document (not a JSON object!). Note that this action should only be carried out when the request has the “finished” status. Otherwise a JSON object will be returned, giving its status.

Example:

```
GET / translation/kim@clang.com/BkY...AE=/08-02-2012%2017:36:46%20CET/11 HTTP/1.1
Content-Type: multipart/form-data
User-Agent: Java/1.6.0_22
Host: bts1.bologna-translation.eu:12344
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive


HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Date: Wed, 08 Feb 2012 16:36:47 GMT

<html>
 <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8
...
</body>
</html>
```

2.6 Cancel an asynchronous request

A running translation request can be cancelled by calling the `DELETE` action on URL:

```
https://{HOST}/translation/{username}/{secret}/{requestTime}/{asyncId}
```

This will return a JSON object with the result:

```
{"asyncId":5,"message":"Request was cancelled.","status":"CANCELLED_BY_USER"}
```

Example:

See 2.4 but use a `DELETE` action instead of a `GET`.

2.7 Synchronous translation requests

Small files can also be translated synchronously, by using the following URL:

```
https://{HOST}/translation/
    sync/{username}/{secret}/{requestTime}/{fileType}/{sourceLanguage}/{targetLanguage}/{domain}
```

The data that is posted should be made up of a multi-part message where the document to translate is stored in a body-part called “content”.

This will return the translated file when it is completed (make sure you set the connection time-out high enough to allow the translation to take up some time). It will not return a JSON object containing an asynchronous ID.

Example:

```
POST /sync/kim@cl.com/Xcvhn...Ffw=/08-02-2012%2017:43:45%20CET/html/DUT/ENG/Legal HTTP/1.1
Content-Type: multipart/form-data; boundary=Boundary_1_1290662793_1328719426109
MIME-Version: 1.0
User-Agent: Java/1.6.0_22
Host: bts1.bologna-translation.eu:12344
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Transfer-Encoding: chunked

81
--Boundary_1_1290662793_1328719426109
Content-Type: application/octet-stream
Content-Disposition: form-data; name="content"


6c8d
<html>
 <head>
.....
  </table>
 </body>
</html>

--Boundary_1_1290662793_1328719426109--

0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Date: Wed, 08 Feb 2012 16:45:08 GMT

2000

<html>
 <head><meta http-equiv="Content-Type" conte
...
  </table>
 </body>
</html>

0
```

2.8 Creating the “secret”

The Secret parameter consists of a string that is used to authenticate the client to the Bologna Translation Service. It is created by calculating a keyed hash message authentication code that is a concatenation of the username, a “#”-character, and the request time (which is also posted in the REST request), and hashing this result with a SHA-1 hash.

In Java, for example, this Secret string can be created using `generateSecret`:

```
final static private String HMAC_SHA1_ALGORITHM = "HmacSHA1";

public String generateSecret(String username, String timestamp,
                  String password) {
      StringBuilder sb = new StringBuilder(username)
                                    .append("#")
                                    .append(timestamp);
      return sign_HmacSha1(sb.toString(), password);
}


protected String sign_HmacSha1(String data, String key) {
      SecretKeySpec signingKey= new SecretKeySpec(key.getBytes(),
                                          HMAC_SHA1_ALGORITHM);
      Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
      mac.init(signingKey);
      byte[] rawHmac = mac.doFinal(data.getBytes());
      return Base64.encodeBase64String(rawHmac).trim();
}
```

2.9 Creating the “Request Time”

The requestTime parameter consists of a string that indicates the time the request is created. This time must be expressed using the following convention:

`dd-MM-yyyy HH:mm:ss z`

where `dd-MM-yyyy` identifies the current day, `HH:mm:ss` the current time and `z` the client’s time zone.

A request time should look like:

`20-03-2010 15:22:07 CET`

This value has to match the requestTime that is used to calculate the Secret. If not, access will be denied to the service due to an authorisation error.

In Java, for example, the requestTime string can be created as follows:

```
final private static SimpleDateFormat DATE_FORMAT =
            new SimpleDateFormat("dd-MM-yyyy HH:mm:ss z");

public String generateRequestDate() {
      return DATE_FORMAT.format(new Date());
}
```