Textkernel Sourcebox

Sourcebox Interfacing Guide

Version 3.0.30
2011-02-28

_textkernel

Disclaimer

Textkernel BV provides this publication "AS IS" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

The publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Textkernel BV may make improvements and/or changes in the TextractorServer and TextractorClient API fundamentals described in this publication at any time.

Ownership

The intellectual property of Textractor, the TextractorClient API and all its components belong to Textkernel BV, located in the Netherlands.

Usage

The Textractor software may have been licensed to third parties for use/integration in a particular project, a particular time frame, and for a particular purpose. It is therefore not permitted to use it for any other purpose than described in the associated license agreement. Unauthorized use of this software, reverse compilation, making illegal copies or the reverse engineering of the Textractor algorithms is STRICTLY PROHIBITED.

See the appropriate license agreement for further details regarding usage.

Confidential Information

 This set of documentation contains confidential information of Textkernel BV, and is protected by copyright of Textkernel BV. It is expressly prohibited to wholly or partly copy this documentation and/or to give third parties, or unauthorized persons within your organization access to this documentation, except with prior written consent of Textkernel BV.

Licensing

For more information on licensing Textractor (or components thereof), please contact Textkernel BV, The Netherlands.

website
http://www.textkernel.nl
e–mail
info@textkernel.nl

Feedback

To constantly improve our software and documentation we very much appreciate your comments on any aspect of our work. Please send feedback to service@textkernel.nl

Table of Contents

Summary 1

Introduction 2

Integration of Sourcebox screens 3
- Log in 3
- Process a document 3
- Edit the extraction results 4
- Retrieve the results 5
- Log out 6

Integration without Sourcebox screens 7
- Atomic HTTP Form POST 7
- SOAP Web Interface 8

External Document Retrieval 9
- LinkedIn URL processing 9
- Calling the Process URL service 9

Automatic Code Table Updates 11
- Automatic Code Table update via the Upload page. 11

TRXML Format 12

Troubleshooting Atomic Post interface 13
- Page Not Found. 13
- Geen Toegang. Toegang Geweigerd bij Inloggen. 13
- Foutmelding 13

Using curl for testing 14
- Integration with Sourcebox screens 14

1 Summary

Textkernel's Sourcebox provides document workflow processing capabilities. It is a fully configurable multilingual web application and web service interface to Textkernel's Textractor Enterprise. Sourcebox also contains interfaces to target systems such as ERP and CRM software. This document provides information on building HTTP based integration solutions.

For more information, please contact Textkernel:

info@textkernel.nl
Textkernel B.V.
Nieuwendammerkade 28 a17
1022 AB Amsterdam
The Netherlands
Tel: +31–20–4942496

2 Introduction

This document describes how to integrate Sourcebox, Textkernel's CV–parsing solution, into another application. Sourcebox provides a graphical web interface, for manual processing and editing of CV–extraction results, as well as web services, for interfacing with external systems.

There are two ways to integrate Sourcebox into another application:

1. Including the Sourcebox web pages for editing the extraction results, after which the extraction results are imported as XML. This is described in chapter Integration of Sourcebox screens, or;
2. Excluding the Sourcebox web pages, and directly retrieving the extraction results as XML. Any additional validation or correction steps must be performed inside the integrating application. This is described in chapter Integration Without Sourcebox Screens.

Chapter Integration Without Sourcebox Screens describes an alternative interface to process CV's from external web sites, such as LinkedIn. Chapter Automatic Code Table Updates describes how to automatically update synonym–enriched Code Tables in Sourcebox.

3 Integration of Sourcebox screens

To use the graphical web interface of Sourcebox, for editing of the extraction results after processing a CV, the following sequence of steps must be taken:

1. Log in with valid credentials;
2. Upload and process a CV in any supported source format;
3. Edit the extracted results using the Sourcebox web pages;
4. Download the structured XML output;
5. Log out

Each step corresponds with an HTTP request. In the following sections, these steps are described in more detail.

In the examples below we assume that the Sourcebox web application is already installed and configured for an account and that the application is running on the URL: http://home.textkernel.nl/sourcebox/

In the appendix it is shown, by way of example, how the HTTP calls can be performed with curl, a simple command line utility.

Note: Textkernel can provide a PHP application as a simple reference for implementing the Sourcebox Screen integration.

3.1 Log in

An HTTP client may access the following URL to log in:

```
http://home.textkernel.nl/sourcebox/loginUser.do?account=XXXX&username=XXXX&password=XXXX
```

If this succeeds (i.e., the given account, username and password are validated by Sourcebox), then:

1. Sourcebox returns a new JSESSIONID both as a cookie, and as a meta element in the html header. This is the session id for the rest of the processing steps, and;
2. Sourcebox will redirect to the main screen of the application, namely:

```
http://home.textkernel.nl/sourcebox/home.jsp
```

If the login attempt is unsuccessful (i.e., the credentials are not accepted), then no session id will be supplied, and instead an HTML page will be returned with an error message.

The JSESSIONID parameter's cookie should be stored and returned to the server along with further HTTP requests. If the functionality is provided by your HTTP library, then the JSESSIONID cookie is sent along automatically. An alternative is to provide the JSESSIONID value as part of the URL parameters, namely the parameter *jsessionid*.

Additional parameters may be passed as part of the login URL. These parameters will be available within the application for later use (this is necessary for some set-ups, such as passing a URL which is later used to send the extraction results to).

Note: every login request must be matched by a corresponding log out request in order to close the session. See Log Out.

3.2 Process a document

To process a CV, perform an HTTP POST with content type "multipart/form-data" to the following URL:

```
http://home.textkernel.nl/sourcebox/processDocument.do
```

The CV should be sent as a form input field named *uploaded_file* as in the following input tag definition:

```
<input type="file" name="uploaded_file">
```

The content type must match the type of document, e.g. *application/msword* for Word documents and *application/pdf* for PDF documents.

As a result of processing the CV Sourcebox returns:

1. On success: a redirect to the edit results page: result–view.jsp, or;
2. On failure: an HTML error page explaining the error (e.g. the document format is not accepted, the file is too large, or the parsing engine is not available).

3.2.1 Sending along additional data

It is possible to provide extra information along side the uploaded CV. For example, in a scenario where a client has provided personal information such as name and e–mail address, it is possible to take this information and override any information extracted from the CV. It is also possible to provide additional meta–data that is useful to store together with the extracted CV (for example job application id).

Sending Trxml

Additional data can be sent as a document in TRXML format (which is Textkernel's internal structured format). The values in the TRXML fields will add or override values extracted from the CV document.

The TRXML should be sent as a form input field named *trxml* as in the following INPUT tag definition:

```
<input type="file" name="trxml">
```

The content type of the trxml must be set to *text/trxml*.

Please refer to TRXML Format for a description of the TRXML format.

Sending custom parameters

If it is only necessary to add information to the result but not to override information extracted from the CV, additional parameters can be sent as HTTP request parameters prefixed with:

```
fieldValue.extraInfo.0.
```

For example: `fieldValue.extraInfo.0.source` defines a parameter called "source".

Note: these additional custom parameters can also be provided using the atomic web services (see next chapter Integration without Sourcebox Screens).

3.3 Edit the extraction results

To view and edit the extraction results in the Sourcebox results page, the client's browser must be directed to result–view.jsp.

Visiting the result–view.jsp web page opens the document in the current session. In order to propagate the session id to the client's browser, the value of the JSESSIONID found in the cookie must be supplied as the jsessionid parameter.

```
http://home.textkernel.nl/sourcebox/result-view.jsp?jsessionid=xxxx
```

Where *xxxx* is the value of the jsession cookie.

3.4 Retrieve the results

Pressing the *export/store* button from within the edit pages in Sourcebox should result in the edited extraction results being imported in the integrating application. How this is implemented depends on the *product* that is configured for the account.

There are three strategies to import the result XML in the integrating application:

1. Sourcebox can redirect the user to the integrating application and trigger a download of the result XML. This is the default and requires only a URL configured to redirect to, or;
2. Sourcebox can be configured to upload the result XML to a page on the integrating application. This requires the integrating application to open up a URL to the outside.
3. The integrating application can simulate pressing the store button and download the result XML itself.

3.4.1 Redirect and trigger a download

The purpose of the redirect, when clicking the *export/store* button, is to notify the integrating application that the result is ready for download.

Redirection is configured by specifying a URL of the integrating application, either:

1. As a Model File parameter *redirectOnSave*, or;
2. As an HTTP GET parameter *redirectOnSave* in the *login* request.

Retrieving the result is done by calling HTTP GET on the following URL:

```
http://home.textkernel.nl/sourcebox/getProfile.do
```

This page returns the result XML if it has been exported/stored, otherwise it returns a text string containing: "NOT AVAILABLE".

Please take into account that all actions must be performed with a valid JSESSIONID, either as a cookie or a URL parameter, otherwise an error will be returned.

3.4.2 Upload to the integrating application

The second strategy is to upload the result XML from Sourcebox to the integrating application when the user clicks the *export/store* button. This requires that the product code must be set to *simplexml* for the account.

The URL to which Sourcebox uploads the result is configured, either:

1. As a Model File parameter: *posturl*, or;
2. As an HTTP GET parameter *posturl* in the *login* request.

The integrating application must accept the result XML from Sourcebox on the configured URL as an HTTP POST request with content type *text/xml* and character encoding UTF-8.

3.4.3 Simulate pressing Store-button and download

As an alternative to using the export/store button inside the Sourcebox application, the integrating application can provide a store-button itself and emulate the button press by sending the corresponding action to Sourcebox:

```
http://home.textkernel.nl/sourcebox/storeDocument.do?buttonPressed=Store
```

Subsequently the result can be downloaded with HTTP GET:

```
http://home.textkernel.nl/sourcebox/getProfile.do
```

The limitation of this method is that no user feedback can be presented on storing a button and consequently this cannot be used in combination with the field validation feature.

3.5 Log out

After the the document has been processed, edited and stored, it is imperative to complete the transaction by logging out (otherwise the session will not be closed and under high load the active session limit might be reached).

The log out is performed by a call to the following URL:

```
http://home.textkernel.nl/sourcebox/logout.jsp
```

4 Integration without Sourcebox screens

When the Sourcebox screens are not used for editing of the extraction results, it is not necessary to divide the processing in several steps as described in the previous chapter. All processing can be done in a single (atomic) HTTP request.

There are two ways of calling the atomic HTTP request:

1. As a HTTP POST *multipart/form-data* request, or;
2. As a SOAP 1.2 web interface.

These are described in the following two sections.

4.1 Atomic HTTP Form POST

The CV to be processed must be passed along with login information and any other data to Sourcebox as an HTTP POST multipart stream with content type *multipart/form-data* to the following URL:

```
http://home.textkernel.nl/sourcebox/processAtomicPost.do
```

The request contains at least the following form parameters:

Table 1. required Process Atomic Post parameters

Parameter	Description	Content type
uploaded_file	The CV to be processed.	File document type, e.g.: application/msword, application/pdf ...
account	the account name of the user	text input
username	the username of the user	text input
password	the password of the user	text input

On success, the request returns the templated result XML, otherwise an HTML error page will be returned explaining the error.

4.1.1 Additional parameters

Additionally, the following parameters are supported:

Table 2. optional Atomic Post parameters

Parameter	Description	Content type
trxml	TRXML, as described in section 2.2 process a document.	File text/trxml
agentSuppliedUser	the username of the user under which processed document should be stored	text input
agentSuppliedAccount	the account name of the user under which the processed document should be stored	text input
skipStore	option to disable storing/exporting the document after processing. Useful if a user logs in on Sourcebox at a later time to edit the results.	text input containing true or false (default is false)

The parameters *agentSuppliedUser* and *agentSuppliedAccount* must be valid credentials, and can only be taken into effect if the user supplying them is configured as *TKAgent* in-

side Sourcebox. If a *TKAgent* supplies only *agentSuppliedUser*, then the *account* is not overridden.

If the parameter *skipStore* is set to true, the storing/exporting will be skipped and the string "OK" will be returned in case of successful processing.

Note: Textkernel can provide an example Atomic Post client in several programming languages at your request.

4.2 SOAP Web Interface

Sourcebox also provides its CV processing service as a SOAP 1.2 web service. The SOAP interface is an alternative way to access the Atomic Post functionality described in the previous section.

The API namespace of this service is:

```
http://home.textkernel.nl/sourcebox/soap/documentProcessor
```

The location of the WSDL file is:

```
http://home.textkernel.nl/sourcebox/soap/documentProcessor?wsdl
```

The service provides three alternative methods:

1. processDocument
2. processDocumentAdvanced
3. processURLAtomic (see next chapter)

The *processDocument* method expects the following parameters:

1. *account (string):* The Sourcebox account
2. *username (string):* The username under that account
3. *password (string):* The password for the username
4. *fileName (string):* The name of the file to be processed
5. *fileContent (base64binary):* The file's binary

If the processing is successful; the method will return the templated result XML. Should the parsing mechanism detect an error, it will throw an exception of type ProcessDocumentException, the interface of which provides further detailed information about the error in question. The table below presents the information wrapped by ProcessDocumentException.

Table 3. SOAP exceptions

Exception field	Description	Example
description	The description of the problem encountered	The credentials given at login are not correct. Please make sure that the account, user and password are correct. If the problem repeats please contact your service vendor. For faster service please supply this output XML, the document which caused the error and your credentials.
tkURL	Context URL	http://home.textkernel.nl/sourcebox/
severity	Problem severity	integer range 0–3, 0 means no error
id	Problem ID	Integer id related to the error description.

Note: Textkernel can provide example SOAP clients for several programming languages on request.

5 External Document Retrieval

Sourcebox also offers the possibility to process documents directly from external websites, by specifying the URL instead of uploading a CV to Sourcebox. This provides an alternative to the Atomic Post service and the processDocument SOAP method. If the Sourcebox account used for URL processing is LinkedIn–enabled, LinkedIn public profiles will be retrieved as structured XML documents via the LinkedIn API in contrast to other, non–LinkedIn resources which will be fetched by regular HTTP GET. Using the LinkedIn API requires additional configuration which is described below.

5.1 LinkedIn URL processing

Sourcebox can process LinkedIn profiles identified by the public LinkedIn URL of a candidate. The public profiles are retrieved in the name of another LinkedIn user. Depending on whether the LinkedIn user of the candidate is in the LinkedIn network of the other user, the retrieved profile may contain more or less information.

In order to use LinkedIn URL processing in Sourcebox, it must be enabled for the account (this can be configured in the Account Settings screen).

5.1.1 Configuring a LinkedIn user

Configuring Sourcebox for LinkedIn parsing entails authorizing Sourcebox to use the identity of a LinkedIn user when retrieving candidate profiles. LinkedIn implements the OAuth authorization scheme (http://developer.linkedin.com/docs/DOC–1008). OAuth requires that a user enters his credentials in the LinkedIn website to authorize its use by Sourcebox. Upon successful authentication LinkedIn returns authentication tokens to be used in retrieving the candidates' profiles.

Sourcebox provides a webpage to perform this authentication with LinkedIn:

```
http://home.textkernel.nl/sourcebox/AuthorizeLinkedIn.do
```

This results in two tokens:

1. linkedinaccesstoken
2. linkedinsecrettoken

These tokens can either be stored in the account (by pressing the button *update account with tokens*) or be passed along as parameters to the LinkedIn service.

5.2 Calling the Process URL service

The Process URL service can be called as a stand–alone atomic service using URL parameters or as a request using the SOAP interface.

The stand–alone atomic service is called at the following location:

```
http://home.textkernel.nl/sourcebox/processURLAtomic.do
```

It requires the following parameters:

1. account – sourcebox account enabled for LinkedIn processing.
2. username – sourcebox username.
3. password – sourcebox password.
4. url – a valid, arbitrary URL. In case of a LinkedIn public profile where the LinkedIn API method is indicated to be used for retrieving the input document for processing, the URL should conform to http://www.linkedin.com/pub/u/v/w or: http://www.linkedin.com/in/username. The 'www' part may be replaced by a country code or locale, for instance 'nl', 'fr', etc. Examples: http://www.linkedin.com/pub/

john–smith/18/297/b16, http://nl.linkedin.com/pub/volten–anne–marie/a/81/721 or http://dk.linkedin.com/in/rahner

5 *outputtype* – can be 'target' (Textkernel extraction format), 'pdf' (PDF document for LinkedIn profiles), 'fingerprint' (MD5 hash of the LinkedIn profile XML or PDF, used for detecting changes) or 'none' (storing the document in the Sourcebox database in case of a persistent account and returning 'OK' in case of success or the exception details if an error has occured).

The following are optional parameters:

1 *method* (optional) – can be 'linkedin' (using the LinkedIn API for retrieving the resource) or 'web' (using a regular HTTP client for fetching the URL). Default is 'linkedin', mainly for historical reasons (it's been implemented before 'web'). Note that method 'web' is incompatible with output type 'pdf'.
2 *linkedinaccesstoken* (optional) – the LinkedIn access token which overrides the LinkedIn access token set up for the account, use only in combination with linked–insecrettoken
3 *linkedinsecrettoken* (optional) – the LinkedIn secret token which overrides the LinkedIn secret token set up for the account.
4 *updateaccount* (optional)– if 'true' calling the service will update the account's linkedinaccesstoken and linkedinsecrettoken given as parameters.

Note: This service requires you to be logged out of Sourcebox because it sets up a new session for the duration of the request and closes it afterwards.

5.2.1 Calling the Process URL service with SOAP

The alternative way to call the Process URL service is to use the SOAP web interface as described in the previous chapter.

The SOAP method is *processURLAtomic* and takes the same parameters as the processURLAtomic.do servlet explained above except for the optional parameters. They have to be specified in the *customQueryString* parameter and conform to the URL parameter format,e.g.: `method=web` or `linkedinaccesstoken=abc&linkedinsecrettoken=xyz`

Please refer to the WSDL for further details on the SOAP implementation: `http://home.textkernel.nl/sourcebox/soap/documentProcessor?wsdl`

6 Automatic Code Table Updates

Part of the CV processing pipeline is the mapping of extracted values to customer specific codes. This mapping is defined by code tables and must be configured in Sourcebox.

Code tables can be uploaded manually in Sourcebox when logged in as a manager or administrator user.

Code table updates can also be automatically uploaded in the following ways:

1. *Upload:* Using the HTTP POST interface of the code table upload page. See the section below.
2. *Fetch:* Using the Sourcebox fetching functionality to let Sourcebox download the code table from an external URL (defined in a model file *CodetableUpdates* block). Please refer to the Sourcebox Administrator's Guide.

6.1 Automatic Code Table update via the Upload page.

The step neede to perform a manual code table upload are:

1. Log in with HTTP GET to retrieve a session id from the cookie (See Chapter Log in): http://home.textkernel.nl/sourcebox/loginUser.do
    1. account
    2. username
    3. password
2. Upload the codetable to http://home.textkernel.nl/sourcebox-new/uploadCodeList.do using HTTP Form POST with Content Type *multipart/form-data* with the following parameters:
    1. fieldname; the name of the codetable, for example: “gendercode”, or “countrysymbol”.
    2. uploaded_file; the codetable file input.

A TRXML Format

The TRXML file is an XML file which conforms to the following DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT TextractorResult (Document,DocumentStructure)>
<!ATTLIST TextractorResult user NMTOKEN #REQUIRED>

<!ELEMENT Document EMPTY>
<!ATTLIST Document lang NMTOKEN #REQUIRED>

<!ELEMENT DocumentStructure (ItemGroup+)>
<!ELEMENT ItemGroup (Item+)>
<!ATTLIST ItemGroup
count NMTOKEN #REQUIRED
key NMTOKEN #REQUIRED
>

<!ELEMENT Item (Field+)>
<!ATTLIST Item index NMTOKEN #REQUIRED>

<!ELEMENT Field (Value)>
<!ATTLIST Field key NMTOKEN #REQUIRED>

<!ELEMENT Value (#PCDATA)>
```

Note: the index attributes must all be integers.

Please contact Textkernel for a TRXML file template for your particular business requirements.

B Troubleshooting Atomic Post interface

In this section, some common HTML error messages in connection with atomic posts are clarified:

B.1 Page Not Found.

The page you tried was not found on our server. You may have used an outdated link or may have typed the address (URL) incorrectly.
Solution: Correct the provided URL.

B.2 Geen Toegang. Toegang Geweigerd bij Inloggen.

Solution: Provide a valid account / user/ password combination.

B.3 Foutmelding

Er is een fout opgetreden tijdens het uitvoeren van ProcessAtomicPost.
com.textkernel.txtor_ui.actions.ActionException:

B.3.1 Detail:

Caught FileUpload exception in saveDocument, ProcessAtomicPost: the request doesn't contain a multipart/form–data or multipart/mixed stream, content type header is null.
Solution: Provide a multipart stream with the requested content type when calling processAtomicPost.do.

B.3.2 Detail:

Caught exception in doTextractorPreProcessing: Leeg document.
Solution: Please submit a document in the stream that is not empty.

B.3.3 Detail:

1. Caught exception in doTextractorPreProcessing: Connection Refused.
2. Caught exception in doTextractorProcessing: Connection Refused
3. Caught exception in doTextractorProcessing: (Normalization)Connection refused.
4. Caught exception in StoreDocument: Connection Refused.

Solution: One or multiple services for CV processing might be temporarily down. Try submitting again. If the problem persists, please contact Textkernel support.

B.3.4 Detail:

Caught exception in doTextractorPreProcessing: Preprocessing Engine Error.
Solution: An unsupported document type was submitted.

C Using curl for testing

curl is a free Linux and Windows command line tool to perform HTTP transfers. This tool can be used for testing the Sourcebox interfaces and might ease development.

C.1 Integration with Sourcebox screens

The five step protocol to integrate with the Sourcebox screens, as described in chapter two, can be performed with the following curl commands.

Assume the following credentials are set up for your account:

- account=testaccount
- username=test
- password=xk45fo

1 login

```
curl http://home.textkernel.nl/sourcebox/loginUser.do \
  --data "username=test" \
  --data "account=testaccount" \
  --data "password=xk45fo" \
  --cookie-jar /tmp/cookie.jar
```

2 upload document

```
curl http://home.textkernel.nl/sourcebox/processDocument.do \
  --form uploaded_file=@/home/geerlings/testcv/NL/petra.doc \
  --cookie /tmp/cookie.jar
```

3 Edit the results in the browser: http://home.textkernel.nl/sourcebox/result-view.jsp?jsessionid=XXX, where XXX is the session variable from /tmp/cookie.jar used above. Then press the export/store button.

4 retrieve result

```
curl http://home.textkernel.nl/sourcebox/getProfile.do \
  --cookie /tmp/cookie.jar
```

5 log out

```
curl http://home.textkernel.nl/sourcebox/logout.jsp \
  --cookie /tmp/cookie.jar
```

C.1.1 Integration with Atomic Post interface

The single step protocol to process a document atomically (without using the Sourcebox screens), as described in chapter 3, can be performed with curl as follows:

```
curl http://home.textkernel.nl/sourcebox/processAtomicPost.do \
  --form uploaded_file=@/home/geerlings/testcv/NL/petra.doc \
  --form username=test \
  --form account=testaccount \
  --form password=xk45fo"
```