D3.1.4: OKAPI COMPONENTS FOR XLIFF

Yves Savourel (ENLASO)

Distribution: Public

MultilingualWeb-LT (LT-Web)
Language Technology in the Web

FP7-ICT-2011-7

Project no: 287815

Document Information

Deliverable number:	3.1.4
Deliverable title:	Okapi Components for XLIFF
Dissemination level:	PU
Contractual date of delivery:	September 30th 2013
Actual date of delivery:	October 31th 2013
Author(s):	Yves Savourel (ENLASO)
Participants:	ENLASO Corporation
Internal Reviewer:	Fredrik Liden (ENLASO)
Workpackage:	WP3
Task Responsible:	Cocomore AG
Workpackage Leader:	Clemens Weins (Cocomore AG)

Revision History

Revision	Date	Author	Organization	Description
0	2013/08/29	Yves Savourel	ENLASO	Initial version
1	2013/08/09	Yves Savourel	ENLASO	Implemented feedback on draft 0.
2	2013/09/11	Yves Savourel	ENLASO	Implemented feedback on draft 1.
3	2013/10/24	Yves Savourel	ENLASO	Implemented delivery date in final copy.

Contents

1 Executive Summary

The goal of the deliverable D3.1.4 was to **implement support for the metadata produced by the LT-Web project in the Okapi Framework components related to XLIFF**.

Overall, this sub-task of the project was successful:

- All stated goals have been achieved on time.
- All the planned components have been available on time.
- Several partners of the LT-Web project are using Okapi tools or libraries in their deliverables.
- Several contractors of the LT-Web project are using Okapi tools or libraries in their deliverables.
- ITS 2.0 has gain recognition in the localization and translation communities through both the Okapi developers and end-users groups.
- The work achieved during the LT-Web project has already proven to be a good base for additional work on other ITS-enabled components.

The **latest version of the running applications** can be found here:
https://code.google.com/p/okapi/downloads/list.

An **online summary of this document** is available here:
http://www.opentag.com/okapi/wiki/index.php?title=MultilingualWeb-LT_D3.1.4

See the section "Material Delivered" in this document for additional links. For more details on the actual deliverables, see the section "Goals and Results".

If you wish to try out the deliverables for yourselves, see the section "Testing the Deliverables".

You can find an outline of the business benefits brought by this work in the section "Business Benefits".

Finally, the section "Implementation Details" provides a more in-depth technical description of the deliverables.

2 Business Benefits

Having ITS 2.0 support in the Okapi Framework provides several business benefits:

From the document authors' viewpoint:

- Making the case to provide ITS metadata inside XML or HTML5 documents is now easier to make: There is a set of low-level or high-level components available across platforms, and under an open-source license, that can be used to build processes using such metadata.

From the localization tools developers' viewpoint:

- Java developers can easily take advantage of the ITS Engine to access ITS metadata at the node level in both XML and HTML5 documents. Because this access is low-level, it can be leveraged to construct many types of applications at a higher level. The developers do not have to process or even know much about global and local rules: they have the resolved data at their finger tips with just a few lines of code.
- The ITS-enabled filters also provide a simple way to access ITS metadata along with extracted data. Quite a few in-house tools and commercial applications utilize Okapi's filters. Using these filters, they can now access ITS metadata without any extra development work. From there they can integrate and take advantage of that information in their own workflows.

From the localization tools users' viewpoint:

- Localization tools users can utilize Okapi's applications out-of-the-box to create pipelines taking advantage of the ITS capabilities of many components. For example: add Text Analysis annotations, mark up Terminology, use standardized Localization Quality Issue reporting, perform Domain-sensitive machine translation, and much more.

3 Goals and Results

The objective of this deliverable was to implement the LT-Web metadata (i.e. ITS 2.0) within the Okapi Framework, especially in connection with XLIFF-related components.

3.1 The Okapi Framework

The Okapi Framework (http://okapi.opentag.com/) is a free, open-source and cross-platform set of components and applications written in Java and designed to help with localization and translation processes.

The code is license under LGPL v3; meaning basically that you can use the code in both open-source and commercial products, and that is a non-viral license.

The project uses and promotes open standards such as XLIFF, TMX, SRX and other formats.

The Okapi Framework started initially in 2005 as a .NET product with few people involved. In April 2009 a completely new code in Java was started and moved to Google Code. Since then, a relatively large team of contributors, with a stable core, has been working on the successive releases.

According the open-source directory Ohloh[1], as of September 11th 2013, the Okapi Framework counts about half a million lines of code, it has “a well established and mature code base” that took an estimated 110 person-year effort to develop (COCOMO model).

The Okapi framework is used through both its built-in applications like Rainbow or CheckMate, as well as third party desktop or Web-based products. It touches on many end-users as well developers. Implementing the LT-Web metadata in Okapi provides instant access for many users.

3.2 XLIFF

XLIFF is the *XML Interchange File Format*, an OASIS standard dedicated to help in the exchange of data to translate and localize across a wide range of tools.

At the time the LT-Web project ends the latest version of XLIFF is version 1.2.

The specification for it is available here: http://docs.oasis-open.org/xliff/v1.2/os/xliff-core.html

Note that, while a new version of XLIFF (v2.0) is being defined, the D3.1.4 deliveries are for the version 1.2 and do not support version 2.0 which is not an official OASIS Standard yet.

3.3 Comparison between the Expectations and the Deliveries

The following table provides a side-by-side summary of the initial expectations for the deliverables, and the results achieved.

Table 1 - Goals and Results

Initial goals as stated in the project’s Description of Work document:	**Corresponding results:**
Improved existing Okapi filters (components able to extract to and merge from XLIFF) to implement metadata support when possible.	The XML Filter and the HTML5-ITS Filter implement support for all relevant ITS data categories. The OpenOffice Filter implements supports for some data categories.
XLIFF Reader with metadata support.	The XLIFF Filter now implements support for ITS annotations. The implementation follows the

[1] http://www.ohloh.net/p/OkapiFramework

Initial goals as stated in the project's Description of Work document:	Corresponding results:
	ITS/XLIFF Mapping best practices document being produced by the ITS Interest Group.
XLIFF Writer with metadata support.	The XLIFFWriter and XLIFFContent classes provide support of ITS 2.0 metadata. The implementation follows the ITS/XLIFF Mapping best practices document being produced by the ITS Interest Group.
Okapi pipeline steps to add, remove, modify and otherwise manipulate metadata; other necessary components as needed.	Several steps have been updated or created to take advantage of the ITS 2.0 metadata. For example: the Enrycher Step, the Quality Check Step, the Term Extraction Step, etc. See the "Steps" section for more details.
The components will be accessible through Okapi tools such as Rainbow, Longhorn and Tikal, as well as directly from other application using the Okapi libraries (including Maven packages) and Web services (e.g. Longhorn clients).	The ITS-enabled components are accessible through Rainbow, Longhorn, Tikal, CheckMate, etc. They are also available as Maven artifacts for any Java application or library that needs them.

In addition to the deliveries provided as part of the project's Work Package 3, the MultilingualWeb-LT Working Group has also initiated the creation of a set of documents describing the best practices regarding how to work with ITS and a number of existing technologies.

One of these documents is a mapping of ITS 2.0 in XLIFF 1.2. The document is being finalized by the International Tag Set Interest Group and is publicly available here: http://www.w3.org/International/its/wiki/XLIFF_1.2_Mapping. The Okapi Framework implementation of ITS in XLIFF follows the recommendations outlined in that document.

3.4 Challenges

Given the large number of ITS data categories ITS 2.0 ended up with and the breath of components they have been integrated with, the deliverable required a lot of work to produce.

A direct effect of this is that only a limited number of pipeline steps in the framework are making use of the ITS metadata (see section "Steps" for the list). There has not been enough time and resource to produce more. However, the Okapi Framework project is continuing to evolve and other pipeline steps will be updated or created to provide yet more functionalities that take advantage of ITS.

3.5 Material Delivered

Downloads of the latest version of the running applications:
https://code.google.com/p/okapi/downloads/list (items starting with "okapi-apps")

General documentation of the Okapi tools and components:
http://www.opentag.com/okapi/wiki

Maven artifacts (releases):
http://repository-okapi.forge.cloudbees.com/release/

Maven artifacts (snapshots, i.e. development version):
http://repository-okapi.forge.cloudbees.com/snapshot/

Google Code project for the Okapi Framework:
https://code.google.com/p/okapi

Source code of the Okapi Framework:
https://code.google.com/p/okapi/source/checkout

Site of the continuous build:
https://okapi.ci.cloudbees.com/

4 Implementation Details

The Okapi Framework is a vast and rather complex set of components. This section aims at outlining the main relevant parts of the code that have been modified or created to support ITS 2.0.

The implementation can be described through several categories:

- ITS Engine
- Helper Classes
- Filters
- Steps

4.1 ITS Engine

The ITS Engine component is the backbone of the ITS support in Okapi. It is a low-level library that allows loading an XML or HTML5 document, applying ITS rules to it, and retrieving the nodes of the document along with the fully processed ITS information.

Every single data category of ITS 2.0, for both global and local rules, for both XML and HTML5, is implemented by the ITS Engine. So any application using this library can easily take advantage of ITS rules and annotations without additional programming effort.

The ENLASO outputs for the ITS 2.0 Test Suite are generated using this component.

The Java documentation for the ITS Engine can be found here: http://okapi.opentag.com/javadoc/org/w3c/its/ITSEngine.html

To try out the ITS Engine see the section “Running the ITS Conformance Tests on a Document”.

4.2 Helper Classes

In order to work with ITS annotation across different parts of the framework, a set of various helper classes are available. They ensure for example a lighter dependency for the components, especially the ones that are not filters.

The main helper classes are the following:

- Annotations Classes
- The ITSContent Class
- The XLIFFContent Class
- The XLIFFWriter Class

4.2.1 Annotations Classes

To carry the ITS metadata after extracting them from the original document, the framework uses annotations that are attached to the different objects corresponding to the mapping of the original format into the Okapi data model. For example, a Storage Size metadata set on an HTML5 <p> element will be represented by a STORAGESIZE annotation on the `TextUnit` object holding the corresponding extracted content.

The `GenericAnnotation`, `GenericAnnotations`, and `GenericAnnotationType` classes handle most of the data categories.

Two specialized classes (`ITSLQIAnnotations` and `ITSProvenanceAnnotations`) also exist to handle the Localization Quality Issue and the Provenance data categories, which require an extended mechanism because of the additional constraints caused by the standoff notation.

The Java documentation for the all these classes are here:
http://okapi.opentag.com/javadoc/net/sf/okapi/common/annotation/GenericAnnotation.html
http://okapi.opentag.com/javadoc/net/sf/okapi/common/annotation/GenericAnnotations.html
http://okapi.opentag.com/javadoc/net/sf/okapi/common/annotation/GenericAnnotationType.html
http://okapi.opentag.com/javadoc/net/sf/okapi/common/annotation/ITSLQIAnnotations.html
http://okapi.opentag.com/javadoc/net/sf/okapi/common/annotation/ITSProvenanceAnnotations.html

4.2.2 The ITSContent Class

The `ITSContent` class is a set of helper methods to output ITS markup in both HTML5 and XML format from given annotations. It also includes common definitions and methods used across components (for example to work with the `annotatorsRef` information, or with the extended language ranges.

The Java documentation for the `ITSContent` class is here:
http://okapi.opentag.com/javadoc/net/sf/okapi/common/filterwriter/ITSContent.html

4.2.3 The XLIFFContent Class

Several components create output in XLIFF format. The `XLIFFContent` class provide for them a common way to generate the XLIFF encoded output of a content (i.e. anything inside the `<source>` or `<target>` elements).

The Java documentation for the `XLIFFContent` class is here:
http://okapi.opentag.com/javadoc/net/sf/okapi/common/filterwriter/XLIFFContent.html

4.2.4 The XLIFFWriter Class

The `XLIFFWriter` class is used by several other components, for example the XLIFF Filter uses parts of it, as well as various pipeline steps creating translation kits. It can also be used on its own to create XLIFF documents from any type of input.

Support for ITS markup is integrated: if one of the objects to output is annotated with an ITS metadata it will be represented properly in the generated XLIFF document.

The Java documentation for the `XLIFFWriter` class is here:
http://okapi.opentag.com/javadoc/net/sf/okapi/common/filterwriter/XLIFFWriter.html

4.3 Filters

The components implementing the IFilter interface in the Okapi Framework allow to extract data and metadata from a wide variety of file formats into a common data model, and to put the modified material back into its original format afterward.

Filters are among the most important components of the framework as they allow a common way to access the content of many different file formats to localize and the metadata that goes along with it.

Several filters include support for different ITS data categories:

- The XML Filter
- The HTML5-ITS Filter
- The XLIFF Filter
- The OpenOffice Filter

4.3.1 The XML Filter

The XML Filter already implemented ITS 1.0. However, with the addition of HTML5 and the large number of new complex data categories in 2.0, this filter has been largely re-coded.

The new implementation uses a base filter in common with the HTML5-ITS Filter. It still supports ITS 1.0 and is backward compatible. It implements most of the ITS 2.0 data categories.

The configuration files used for the XML Filter are in ITS format with a few additional options encoded in a separate namespace.

The user documentation for the XML Filter is here:
http://www.opentag.com/okapi/wiki/index.php?title=XML_Filter

4.3.2 The HTML5-ITS Filter

The Okapi Framework offers several ways to process HTML document. The new HTML5-ITS Filter provides direct ITS support in HTML5 documents. The filter implements most of the ITS 2.0 data categories.

This filter uses a base filter in common with the XML Filter and adds the HTML5-specific handling. This is made possible by using the `HtmlDocumentBuilder` class of the Validator.nu HTML Parser, an open source library that can reads valid HTML5 documents into an XML DOM object.

The user documentation for the HTML5-ITS Filter is here:
http://www.opentag.com/okapi/wiki/index.php?title=HTML5-ITS_Filter

4.3.3 The XLIFF Filter

In the Okapi Framework reading an XLIFF document is done the same way as any other type of document: through a filter, in this case the XLIFF Filter. The data extracted from the input document are mapped to the internal Okapi data model which includes ITS support.

Because XLIFF is a format dedicated to localization, it has many features that overlap the ITS data categories, partially or completely. To allow the XLIFF documents to be handled properly by both an XLIFF processor and an ITS processor the document is expected to use a specific representation of the ITS data categories. The representation uses XLIFF constructs when it makes sense and native ITS markup in other cases.

The representation is defined by the ITS Interest Group and is described here:
http://www.w3.org/International/its/wiki/XLIFF_1.2_Mapping.

The user documentation for the XLIFF Filter is here:
http://www.opentag.com/okapi/wiki/index.php?title=XLIFF_Filter

4.3.4 The OpenOffice Filter

The `ODFFilter` class that is used in the OpenOffice Filter to process ODF files implements the local notation of the following data categories: Translate, Localization Note, Terminology and Locale Filter.

Entering the ITS information in the ODF document can be done different ways, for example by using the plug-in developed by]init[for LibreOffice. See http://www.init.de/en/libreofficeWriter for more details.

The user documentation for the OpenOffice Filter is here:
http://www.opentag.com/okapi/wiki/index.php?title=OpenOffice_Filter

4.4 Steps

Several pipeline steps have been updated or created to take advantage of ITS annotations.

One of the main advantages provided by the Okapi Framework is that steps are working on the extracted data and therefore you can apply the ITS features to any content support by the many Okapi filters, even if that format does not support ITS.

You can find a list of the filters available here: http://www.opentag.com/okapi/wiki/index.php?title=Filters

The main steps that are using ITS annotations are currently:

- The Enrycher Step
- The LanguageTool Step
- The Microsoft Batch Translation Step
- The Quality Check Step
- The Term Extraction Step

4.4.1 The Enrycher Step

This step implements access to the Enrycher Web service provided by the Artificial Intelligence Laboratory of the Jožef Stefan Institute.

This service allows to annotate an HTML5 input with information coded using the ITS Text Analysis data category (This is the deliverable D3.1.3 of the project).

You can see a presentation of the service here: http://ailab.ijs.si/tools/enrycher/, and a Web page to access the service manually is available here: http://enrycher.ijs.si/.

The Enrycher Step allows annotating any extracted content generated by one of the Okapi filters. You can for example apply the service to Word DOCX or OpenOffice documents, FrameMaker MIF files, Java properties resources, etc. This works by converting temporarily the internal extracted okapi data into HTML5, sending it to Enrycher, and converting the data back.

Once annotated, the extracted content can go through other steps providing other actions. For example, the Term Extraction Step can product a list of term candidates from the Text Analysis markup. Another example is the creation of a translation kit in XLIFF where the annotated content is directly available to the translators.

The user documentation for the Enrycher Step is here: http://www.opentag.com/okapi/wiki/index.php?title=Enrycher_Step

4.4.2 The LanguageTool Step

The LanguageTool project (http://www.languagetool.org/) is an open source effort to provide a strong mechanism to verify style, grammar and other linguistic aspect of various contents, in various languages. The potential problems identified by the system support the types defined for the ITS Localization Quality Issue data category.

The LanguageTool Step applies the verifications done through the LanguageTool library to any extracted content generated by one of the Okapi filters and encodes the results using Localization Quality Issue annotations.

The user documentation for the LanguageTool Step is here: http://www.opentag.com/okapi/wiki/index.php?title=LanguageTool_Step

4.4.3 The Microsoft Batch Translation Step

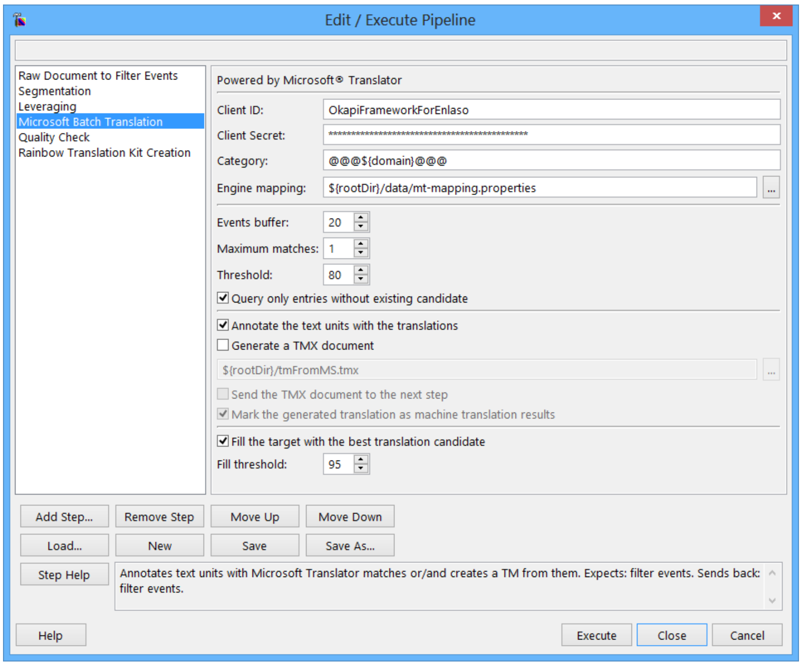
The Microsoft Batch Translation Step provides a way to send any extracted content generated by one of the Okapi filters to Microsoft Translator Web services and obtain back translation candidates.

The ITS-related aspect in this step is the option of selecting, through the ITS Domain data category, which system deployed in Microsoft Translator Hub to use.

When the `Category` field of the step's parameters is set to `${domain}`, this variable is replaced by the first occurrence of the value for the Domain annotation found on an extracted text unit.

The different systems deployed on Microsoft Translator Hub are identified with auto-generated GUIDs. The mapping between the domain value and the system GUID to use is done by: a) placing the variable between a pair of @@@ strings and b) a lookup table defined in the `Engine mapping` field of the parameters.

Figure 1 - The parameters of the Microsoft Batch Translation Step



Raw Document to Filter Events
Segmentation
Leveraging
Microsoft Batch Translation
Quality Check
Rainbow Translation Kit Creation

Powered by Microsoft® Translator

Client ID: OkapiFrameworkForEnlaso
Client Secret: **
Category: @@@${domain}@@@
Engine mapping: ${rootDir}/data/mt-mapping.properties

Events buffer: 20
Maximum matches: 1
Threshold: 80

☑ Query only entries without existing candidate

☑ Annotate the text units with the translations
☐ Generate a TMX document
${rootDir}/tmFromMS.tmx
☐ Send the TMX document to the next step
☑ Mark the generated translation as machine translation results

☑ Fill the target with the best translation candidate
Fill threshold: 95

Annotates text units with Microsoft Translator matches or/and creates a TM from them. Expects: filter events. Sends back: filter events.

The documentation for the Microsoft Batch Translation Step is here: http://www.opentag.com/okapi/wiki/index.php?title=Microsoft_Batch_Translation_Step

4.4.4 The Quality Check Step

The Quality Check Step let you apply a wide variety of verifications to the source and/or target content in the Okapi data model. ITS 2.0 is involved at different level here:

- Existing annotations for the Localization Quality Issue data category are harvested and integrated into the step's report.
- Existing annotations for the Storage Size or the Allowed Characters data categories are taken into account and corresponding issues are generated if needed.
- New issues created by the step are attached to their corresponding content using the annotation for the Localization Quality Issue data category.

The step can be used in a pipeline, or from within the CheckMate application.

The documentation for the Quality Check Step is here: http://www.opentag.com/okapi/wiki/index.php?title=Quality_Check_Step

4.4.5 The Term Extraction Step

The Term Extraction step provides a way to generate a list of term candidates from any extracted content generated by one of the Okapi filters.

The step allows spans of content annotated for the Terminology or for the Text Analysis data categories to be seen as potential term candidates and extracted.

The user documentation for the Term Extraction Step is here:
http://www.opentag.com/okapi/wiki/index.php?title=Term_Extraction_Step

5 Testing the Deliverables

The Okapi Framework is vast and complex; the components that are ITS-enabled are only a small part of the overall libraries and tools. This section aims at providing some help on how to test some of the ITS-aware components

5.1 Installing the Okapi Tools

Download the latest release of the okapi-apps distribution for your platform (Windows, Linux and Macintosh are supported for both 32 and 64-bit). It can be found here: https://code.google.com/p/okapi/downloads/list.

How to install the tools is described in the `readme.html` file included in the distribution. Normally you simply need to un-zip the files in a new directory and you can get started.

The following examples assume that:

- You are running under Windows (but this can easily be adapted for the other platforms).
- You have installed you Okapi tools under `C:\OkapiTools` (and this directory is referred to as the *installation directory* later in this document)

5.2 Running the ITS Conformance Tests on a Document

This example show how to check if the ITS Engine process properly an XML or HTML5 document.

1. Open a DOS prompt
2. Go to your installation directory
3. Enter the following command:

```
C:\OkapiTools>itstest examples\myFile.html
```

This will generate a new file named `myFileoutput.txt` in the `examples` directory. This file is in the test output format used by the conformance test suite. Each line shows an XPath location corresponding to a node of the document, and the resolved ITS information for the data category tested.

You can use `itstest.bat` to create the test output for any XML or HTML5 documents with or without ITS markup. By default it create the file for the Translate data category, but you can choose other data category with the `-dc` option. For example:

```
C:\OkapiTools>itstest examples\myFile.html -dc withintext
```

This will create the test results for the Element Within Text data category. Use the `-l` option to list the keyword to use for each data categories.

If needed, you can also use the `-r` option to provide an external global rules file.

5.3 Extracting XML and HTML5 Documents to XLIFF

This example uses the XML Filter, the HTML5-ITS Filter, and the helper classes.

- Go to the installation directory
- Double-click on `startTikalPrompt.bat`. This will start a DOS prompt window from where you can use Tikal.
- Go to the `examples` sub-directory of the installation directory.
- Enter the following command:

```
C:\>OkapiTools\examples>tikal -x myFile.html -fc okf_itshtml5
```

This will create an XLIFF document named `myFile.html.xlf` that contains the data extracted based on the defaults and stated ITS information available for the HTML source, as well as any ITS metadata that can be mapped to XLIFF and useful during the localization process.

By default the XLIFF document targets French as the translation. But you can use the option `-tl` and the BPC47 language code of your target. You can also use the option `-sl` the same way to specify another language than English as the source language.