

**5D2.2: Initial L3Data Schema and Architecture**

**David Lewis, Leroy Finn, Kevin Koidl,**

**Distribution: Public Report**

**Federated Active Linguistic data CuratiON (FALCON)**

FP7-ICT-2013-SME-DCA

Project no: 610879

## Document Information

|  |  |
| --- | --- |
| **Deliverable number:** | D2.1 |
| **Deliverable title:** | Initial L3Data Schema and Architecture |
| **Dissemination level:** | PU |
| **Contractual date of delivery:** | 31st March 2014 |
| **Actual date of delivery:** | 28th April 2014 |
| **Author(s):** | David Lewis, Leroy Finn, Kevin Koidl |
| **Participants:** | Interverbum, TCD |
| **Internal Reviewer:** | DCU |
| **Workpackage:** | WP2 |
| **Task Responsible:** | TCD |
| **Workpackage Leader:** | Interverbum |

## Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Organization** | **Description** |
| 1 | 3/3/2014 | David Lewis | TCD | Draft of scHema |
| 2 | 28/04/14 | Leroy Finn | TCD | API Design |
| 3 | 8/4/14 | Leroy Finn | TCD | Typos corrected |
| 4 | 27/4/14 | David Lewis | TCD | Final edits |

Contents

Document Information 2

Revision History 2

1. Executive Summary 5

2. Introduction 5

3. Platform Architecture 6

4. L3Data Schema Design 7

4.1. Reuse of W3C Provenance as a Process Analytics Model 8

4.2. Reuse of the Natural Language Processing Interchange Format (NIF) 10

4.3. Reuse of CNGL Global Intelligent Content Model 10

5. L3Data Architecture Design 12

5.1. Initial L3Data architecture 13

5.2. Initial L3Data Functionality 13

Registering L3Data Components 14

Adding file/files to Alfresco 14

Provenance Generation 15

5.3. Initial CMIS Model 15

What is a CMIS object? 15

L3Data CMIS Model 15

5.4. Initial RESTful service 16

Registering L3Data Components 16

Provenance Generation 17

Provenance Data Retrieval 18

5.5. Initial Provenance Logging 19

Prov Logger Overview 19

Initial Provenance Logging Overview 19

Initial Provenance logging algorithm 20

Updating the XLIFF to RDF transformation 20

Initial schema for resource identification 21

Technologies used 21

5.6. Initial L3Data schema mapping 21

6. Conclusions and Next Steps 22

1. Executive Summary

This is the initial design documentation that drives the first cycle of development and that provides a baseline to evaluate subsequent cycles for L3Data schema and architecture. This design is based on initial use case requirements capture performed by the FALCON project[[1]](#footnote-1). These form part of the L3Data Federation Platform, a language resource sharing platform using linked data to manage the active curation and sharing of language resources relevant to language technology support over multi-party localisation value chains.

1. Introduction

We follow the decentralised evolution principles of open linked data, in making use of and extending existing RDF vocabularies, specifically making use of existing vocabularies specifically the W3C Provenance Model ontology[[2]](#footnote-2) and emerging linked data vocabularies for language resources and the ontology designed for the conversion of the International Tag Set 2.0 into RDF[[3]](#footnote-3).

A content processing provenance approach is taken to the core capability modelling of the L3Data Schema, where the PROV is used to express the state transformation that operates on content (principally on documents, terms, translation units and segments) and its metadata. These transformations results from localisation content processing activities performed by the human, automation/tool and organisations involved, the activity type, and the activity timing. It can also include activity specific attributes such as: confidence scores; quality assessments; fine-grained interaction (e.g. keystroke) logs from tools; license/copyright terms and links to other web resources, e.g. term bases or ontologies. The activity typing supports: authoring/revision; translatable content extraction and segmentation; source quality assurance (including terminology usage); Translation Memory leverage; SMT usage; human translation; post-editing and target language quality assurance. This provenance-based approach can also be used to record value-adding activities applied to shared translation memory, term-base or parallel text elements, such as adding term translations, definitions or morphologies or identifying terms, or style and domain classification of entries. As this provenance-based approach serves to capture both the activity resulting in the recorded content transform or annotation and the agents responsible, it thereby provides the basis for the management of acknowledgement and credit for shared language resources, enabling auditing and ROI calculation, as well as supporting license/copyright based access control through executable policy rules.

Modelling will follow an evolutionary path, revised over the two evaluation cycles based on feedback on system integration experiences, data handling performance measures and the functional sufficiency of the model in evaluations performed in the FALCON project. It will also be revised in response to relevant external usage and feedback on published version of this specification and from usage experiences of the publically available portal that the FALCON project is developing based on this model.

This document provides therefore an initial system architecture which is designed to provide a durable framework for developing the L3Data Federation Platform as well as for leveraging standards for integration between industry partner software platform and for the future integration of the Platform with industry tools beyond the scope of the FALCON project. The Architecture will leverage TCD’s experience in system integration based on linked data and XTM’s experience in developing standards and implementations for its web service based localisation tool bench. The architecture allows L3Data for an individual translation project to be both distributed and interlinked between separate L3Data stores operated by different language resource providers, LT service operators (SMT or NER) or SaaS web tool operators (for SME tools). The integration of this architecture between existing localisation tools provides a demonstration of the L3Data Federation Platform in representative localisations workflow, and thereby validates and enables the revision of different parts of the Architecture and Schema.

The upload of new L3Data and inter-store links as well as the searching, filtering and retrieval of L3Data will be supported in the Architecture through the definition of RESTful Web APIs which wrap SPARQL expression in task specific operations related to the localisation activities supported. The API will be structured to support different types of usage:

1. for publishers of L3Data, allowing them to import language resources from other open formats (e.g. CSV, TBX, TMX, XLIFF), associate provenance and license information with these resources, set access control rules and track standoff linking to these resource;
2. for workers in localisation workflows that employ L3Data-enabled tools, allowing them to search multiple L3Data stores using SPARQL query templates parameterised to their needs via the API. It will also allow them to make linking associations between L3Data, e.g. associating the multiword unit in a bi-text segment with a term record. This API will be used by the workflow analysis tools to be included in the platform;
3. for tool integrators (both within the project and, subsequently, third parties), to allow efficient and more customisable access and control over L3Data, including support for better performance in accessing large volumes of data and the integration of access control setting with native tool settings.

The API will be specified as part of the Architecture.

1. Platform Architecture

The L3Data Platform Architecture aims to support the use of linguistic linked data in localisation workflows. It aims to work in harmony with existing best practice in interoperable localisation workflows. Principally this involves the use of XML Localization Interchange File Format (XLIFF)[[4]](#footnote-4), which is standardised by OASIS, to communicate between localisation components developed by different vendors. Such XLIFF-based interoperability can operate between tools obtained by different vendors but operated with in a single LSP domain, or it can operate between different services providers in more complex localisation value chains. XLIFF allows exchange of content extracted into translatable segments, typically at the sentential level.

The Architecture leverages this interoperability by using XLIFF as the main source of content and meta-data in localisation workflows. An XLIFF proxy pattern is used to analyse XLIFF files exchanged at different points in the workflow and to build up a provenance model in RDF of the changes made to the content being translated and the associated workflow meta-data. This means that RDF provenance model can be generated without requiring major reengineering of existing, standards-based localisation tool chains.

We focus on the integration of language technology into localisation workflows, in particular machine translation and text analysis technology that can support terminology identification. Though XLIFF is suitable for passing content to such components, e.g. as established by the M4LOC XLIFF encapsulation of the Moses SMT platform[[5]](#footnote-5), it does not support the exchange of meta-data that can be usefully be consumed and generated by these components. However, the Internationalization Tag Set 2.0 (ITS2.0) Recommendation recently standardised by the Multilingual-Web Language Technology working group at the W3C[[6]](#footnote-6) does provide appropriate meta-data standards, addressing meta-data related to text annotation and terminology annotation of word and phrases, domain and confidence scores related to machine translation of text, and quality assurance annotation for source and target text. ITS2.0 is specified as a meta-data annotation for XML or HTML5 documents, and hence can be readily applied to XML interoperability formats already in use in localisation tools. As there is some overlap between ITS2.0 and XLIFF, a mapping for use of ITS2.0 in XLIFF1.2[[7]](#footnote-7) has been specified via a collaboration between the MLW-LT WG and the XLIFF Technical Committee.

In addition to XLIFF, other interoperability formats are important in localisation workflow and are therefore addressed by the architecture. These include:

* Profiles and extensions of XLIFF 1.2: These include profiles used by major LSPs such as SDL and the XLIFF:doc profile[[8]](#footnote-8) from the Interoperability Now! (IN!) group[[9]](#footnote-9). XLIFF itself is also being revised as XLIFF2.0[[10]](#footnote-10).
* Translation Memory eXchange (TMX)[[11]](#footnote-11), for exchanging translation memories between tools.
* Term Base Exchange (TBX)[[12]](#footnote-12), for exchanging terminology bases between tools.
* Linport: The Language Interoperability Portfolio Project[[13]](#footnote-13) has produced a specification[[14]](#footnote-14) that captures important parameters of localisation projects.
* Translation Interchange Package Protocol (TIPP)[[15]](#footnote-15) which provides a container format for collecting elements related to a translation job, including XLIFF file, Linport documents, translation memories and term bases.

As the TIPP captures much of the current best practice in localisation interoperability, the L3Data architecture operates a proxy that will capture and log not only the project contents and meta-data from an XLIFF file, but also that from a TIPP package to provide appropriate context for managing the sharing, access control and quality assessment of localisation language resources as linked data.

The following sections explain both the initial L3Data Schema (in the form of a linked data vocabulary) and the initial Platform Architecture in terms of the APIs involved in generating and accessing instances of the L3Data Schema within a localisation workflow.

1. L3Data Schema Design

The L3Data Schema design follows best practice in leveraging existing vocabularies wherever possible and appropriate. The Schema therefore includes elements of the following existing linked data vocabularies:

The Internationalization Tag Set (ITS) 2.0 vocabulary which defines annotations for the textual content of HTML and XML contents that related to the internationalisation and localisation of

NLP Interchange Format (NIF) which defines a way of representing Strings in RDF, linking those into structure like phrases, sentences and paragraph and to provide annotations and linking structures related to natural language processing such as part of speech.

Provenance Ontology which is W3C recommendation which provides a means for recording the proveance of activities enacted on entities by specific human or automated agents

The Global Intelligent Content (GLOBIC) vocabulary was developed by the CNGL Centre for Global Intelligent Content[[16]](#footnote-16). This addresses common activities and entities for a range of content processing types that relate to the use of a range of language technologies. This model extends the W3C Provenance Ontology in order to provide a minimal common model for the end-to-end monitoring complex content processing chains. Only the portion of the GLOBIC model that applied to multilingual content processing is used here.

## Reuse of W3C Provenance as a Process Analytics Model

A primary objective of the Schema is to enable end-to-end analytics across localsiation processing chains consisting of heterogeneous content processing components. This requires a model that records the execution of processes with the content that is being processes and the agents that were responsible for the execution of those processes. Such a semantic model is available in the form of the [W3C Provenance Ontology](http://www.w3.org/TR/prov-o/), which is an RDF binding of the [Provenance Data Model](http://www.w3.org/TR/prov-dm/).

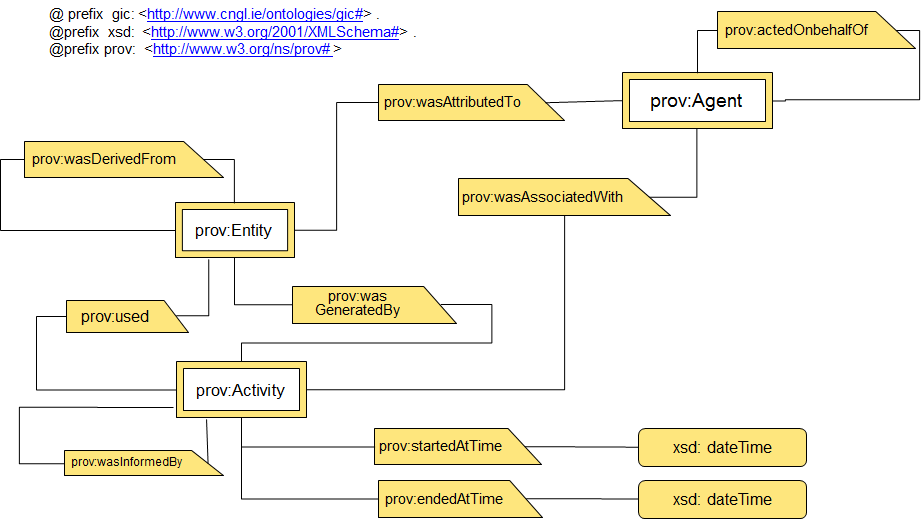


Figure 1: The starting point subset of the W3C Provenance Ontology (PROV-O)

This model is selected therefore as the common basis for the recording of process chain execution, thereby providing the basis for end-to-end analytics and localisation processing path processing rules. In addition, such a common record will also provide a basis for harvesting natural language corpora of various types as it is generated in content processing chains. Such active curation this content is key to retrain and therefore improve the performance of intelligent content processing components in a processing chain, both individually or in concert.

Key outcomes in terms of the ‘style’ of the semantic model of adopting the PROV-O model are:

* Models focus identifying classes that are sub-classes of prov:Entity, prov:Activity and prov:Agent and relationships between them reuse or specialise existing PROV-O properties.
* New properties should be conceived in the past tense, e.g. wasTranslatedBy.
* As a result, Schema aims to be instantiable with the aim of establishing concrete facts resulting from Global Intelligent Content processing to enable the analysis of the process chain and the active curation of it outcomes. The PROV-O model ensure all these facts are captured in terms of specific entities, of the activities that operated on those entities and agents involved in conducting those activities.

The core PROV-O elements reused are described below for clarity.

Class Definitions:

* **prov:Entity** is something the provenance of which is being recorded. The PROV-O specification defines it as: a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.Primarily Entities in the GIC model will be Content or Metadata associated with Content.
* **prov:Activity** is a process that generated or operated upon an Entity in a way that is recorded in a provenance model. The PROV-O defines an activity as: something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.
* **prov:Agent** is a person, tool or organisation involved in conducting an Activity that is recorded as impacting on the provenance on an Entity. The PROV-O specification defines an agent as something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

Property Definitions taken directly from [PROV-O](http://www.w3.org/TR/prov-o/):

* **prov:wasDerivedFrom**: A derivation is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity.
* **prov:wasAttributedTo**: Attribution is the ascribing of an entity to an agent
* **prov:wasAssociatedWith**: An activity association is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity.
* **prov:wasGeneratedBy**: Generation is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation.
* **prov:wasInformedBy**: Communication is the exchange of an entity by two activities, one activity using the entity generated by the other.

The above class and property definitions constitute a [starting point](http://www.w3.org/TR/prov-o/#description-starting-point-terms) set of the PROV-O ontology. The ontology also supports [expanded](http://www.w3.org/TR/prov-o/#description-expanded-terms) classes and properties that specialise the starting point and a set of [qualified](http://www.w3.org/TR/prov-o/#cross-reference-qualified-terms) classes and properties that allow further attributes to be expressed for the binary relations in the starting point and expanded properties. While the full set may be found in [PROV-O](http://www.w3.org/TR/prov-o/), below we present a selected subset of particular relevance to the use of PROV-O for Global Intelligent Content.

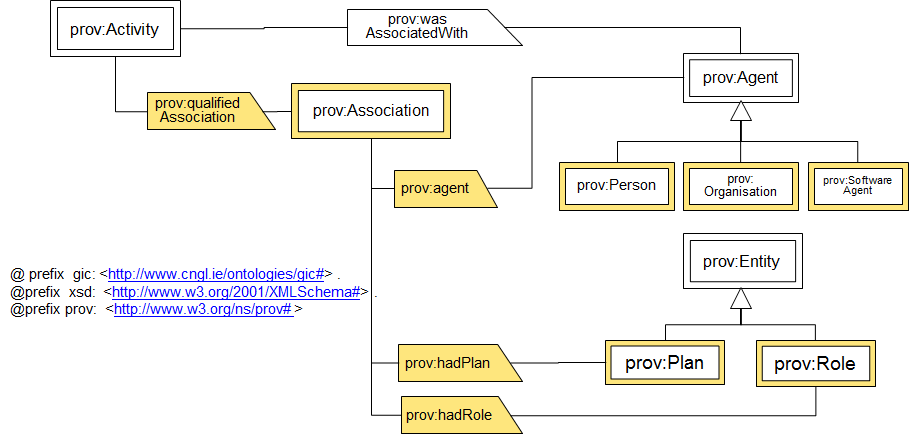


Figure 2: Selected relevant parts of the expanded and qualified concepts and relationship from PROV-O

A prov:wasAssociatedWith property can be enriched with a Qualified Association property. This links to an Association class, which both indicates the agent involved, thereby identifying the property it qualifying and also allows qualification by a Plan and a Role. These are defined as:

* **prov:Plan** is an entity that represents a set of actions or steps intended by one or more agents to achieve some goals. PROV-O does not elaborate on the nature of Plan. In the context of FALCON a Plan may for example represent an template path for content between Content Processing Components (e.g. specified in a workflow model), constituting a localisation workflow
* **prov:Role** is the function of an entity or agent with respect to an activity, in the context of a usage, generation, invalidation, association, start, and end.

## Reuse of the Natural Language Processing Interchange Format (NIF)

In capturing the provenance of textual content and its meta-data based on existing XML-based interoperability standards, e.g. XLIFF, ITS2.0 and TIPP, the Schema follows the approach taken by the Natural Language Processing Interchange format (NIF v2.0)[[17]](#footnote-17). This has defined a linked data vocabulary[[18]](#footnote-18) that can represent Strings a subjects of RDF triples and therefore allows them to the further annotated using RDF predicates. It also provides a way of linking these strong back to their source documents. In the L3Data Architecture, the specific approach for extracting strong and their annotations from ITS2.0 annotated content, as specified in the ITS2.0 specification[[19]](#footnote-19), is adopted.

## Reuse of CNGL Global Intelligent Content Model

Here we describe the concept from the CNGL Global Intelligent Content Model that are reused as part of L3Data Schema.

The concepts in the Schema align closely with the World Wide Web (W3C) [Web Architecture](http://www.w3.org/TR/webarch/) in that the main concept is a **Resource** that can be identified by a URI. A Resource can refer to other Resources using hyperlinks. There are two subclasses of Resource again in line with the Web Architecture. One is InformationResource which is further specialised into Data and Content. **Content** *is a Resource created by a person with the purpose of it being consumed by other people.*

**Data** in being identified by a URI, is consistent with the notion of the [Semantic Web](http://www.w3.org/standards/semanticweb/) and [linked data](http://www.w3.org/standards/semanticweb/data). However, in this model it can refer to any Information Resource that is not deemed to be Content, but plays a role in content processing.

The other Resource subclass is the **Service**, which is consistent with the W3C Web Services Architecture.

Central then to the Schema is the role of a **Content Processing Component**. This takes on two roles in relation to the Provenance model. The first is when a Content Processing Component is an actor associated with a **Content Processing Activity.** This is a generalisation of any activity that involvesprocessing of content, so it both uses Information Resources and Information Resources are generated by it. More specifically the activity involves the *use of a Service* that is *offered by* the Content Processing Component. The role role taken by a Content Processing Component is as an Entity that, along with the Services it offers, is generated by a **Component Generation Activity.** This activity is associated with a specialised Agent called a **Component Factory** and uses Information Resources (typically training corpora) to generate Content Processing Components, e.g. a machine translation engine.

The following classes and property definitions address the core entities and activities subclassed from the W3C PROV-O vocabulary and relevant to the FALCON use cases.

Prov:Entity subclass definitions as follows. Each except Project and Quality Assessment are subclasses of nif:String:

* **Project** is the representation of a localisation project within an organisational domain. It is annotated with the details and resources typically found in a TIPP manifest, including the project meta-data expressed in a Linport descriptor.
* **SourceUnit** is an a string extracted from a source document for the purposes of translating the text it contains;
* **Segment** is a string resulting from subdividing source content to a granularity designed to ease translation. Typically this is a sentence, but may also be text from a heading cell or a table;
* **Term** is a string that is subject to terminology management with the intention of supporting consistent source content and the consistent translation of the source term;
* **Translation** is a string that was produced by translating another string from its source language into a specified target language;
* **TranslationRevision** is the revision of a string that was originally a Translation;
* **Quality Assessment** is a linguistic assessment of content that highlight errors and other quality issues including those related to content that was a translation.

Property Definitions:

* **wasTranslatedFrom** is a sub-property of prov:wasDerivedFrom that links a Translation (as range) to the corresponding source text. This source could be a Segment, or a Term.
* **wasAssessmentOf** is a sub-property of prov:wasDerivedFrom that links some Content to a Quality Assessment.

Prov:Activity subclass definitions are as follows:

* **Text Analysis** is the annotation of some portion of text, typically a word or phrase, with some lexical or terminological meta-data;
* **Extraction** is process of extracting translatable text from some source content for the purposes of translation;
* **Segmentation** is the subdividing source content to a granularity designed to ease translation.
* **Translation** is the translation of any source content, including Source Unit, Segment, Term, Word or Phrase into a target language;
* **Translation Revision** is the change of a translation after consideration of the translation and the corresponding source content;
* **Quality Review** is the assessment of the linguistic quality of a translation considering the translation, the revised translation or the source. It may also be applied to any of the source content.
* **Merging** is the integration of translated content into a completed target language document format.

A full specification of the Schema, including the supporting classes and properties derived from properties of existing specifications are provided in the attached documents.

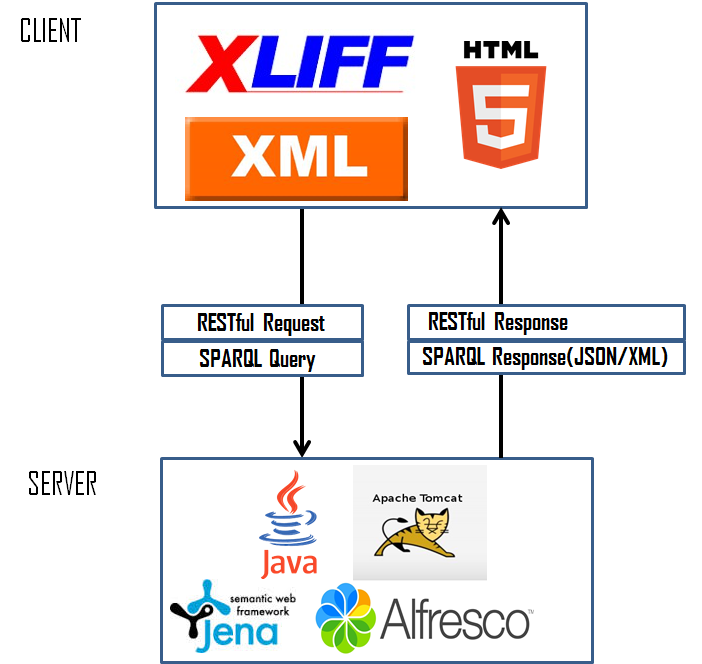
1. L3Data Architecture Design

The L3Data Federation Platform Architecture aims to enable tools in the localisation tool chain to generate and use L3Data in linked data stores that are distributed across more than one organisation domain across the value chain.

To minimise the impact on existing tools, the primary mechanism used for generating L3Data is a proxy mechanism, where localisation workflow data exchanged using existing well defined exchange formats are intercepted and used to log the state of the content and its meta-data using the provenance-based L3Data Schema. If the exchanged format is part of an existing workflow instance, the previous log of that workflow are analysed and compared in order to log just the subsequent *changes* to the content and its meta-data.

Though this document is intended to offer an initial reusable architecture, the work takes an implementation-led approach so as to reduce the risks related to the un-anticipated shortfalls in the compatibility or functional scope of the different formats or of their implementations across different tools. The architecture therefor makes references to specific implementation of platforms that implement different open models or interfaces that are then integrated into the overall architecture. The proxy model consists of clients, which will be existing tools in the localisation workflow that can provide snap-shots of the workflow data. In the FALCON implementation these will consist of XTM Cloud[[20]](#footnote-20), TermWeb[[21]](#footnote-21) and EasyLing[[22]](#footnote-22). The server side captures the content snapshots and associated meta-data as L3Data using RESTful Web APIs designed to ease integration with localisation tools but that encapsulate more complex, open interfaces for logging and sharing both content and linked data. For content the OASIS Content Management Interoperability Service (CMIS) web service[[23]](#footnote-23) is used while for linked data the W3C SPARQL semantic web query language[[24]](#footnote-24) is used.

The exploratory implementations of these APIs have been developed using open source platforms so as to ease the transition to an open source reference architecture. In this initial exploration prototype, Jena has been used for the SPARQL and the linked data store implementation, Alfresco for CMIS and Apache Tomcat for the Web API implementations, as summarised in figure 3 below.

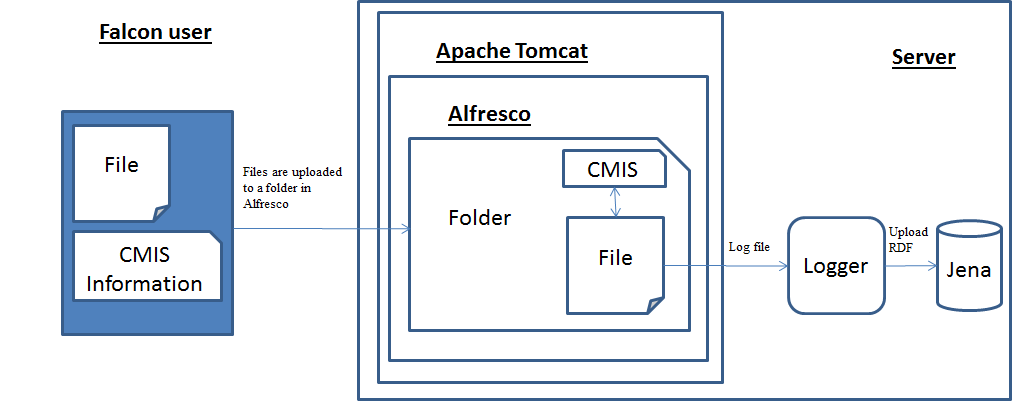


**Figure 3: Initial L3Data technology stack architecture**

The rest of this section will describe the following:

* Initial L3Data system architecture
* Initial L3Data functionality
* Initial CMIS model
* Initial mapping for RESTful service
* Initial Provenance logging
* Initial L3Data schema mapping

## Initial L3Data architecture



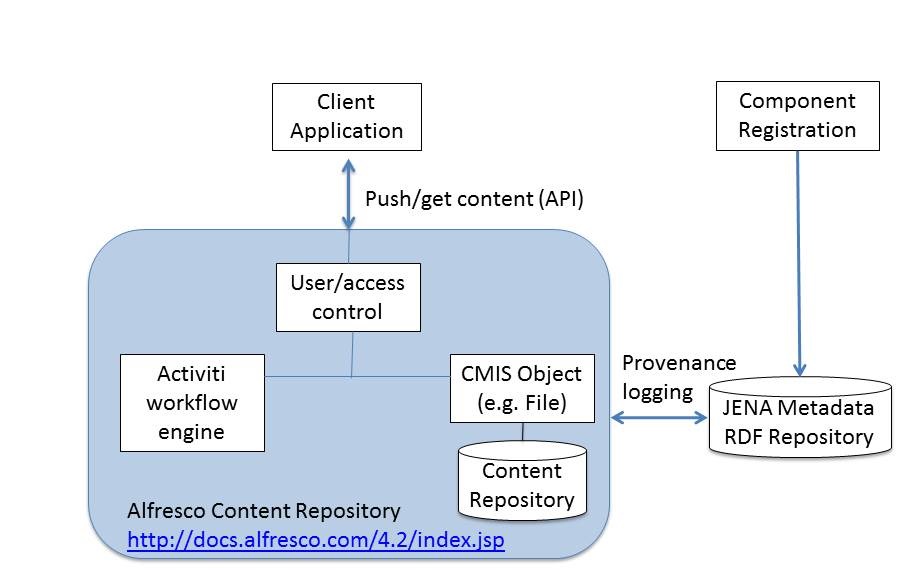
**Figure 4: Initial L3Data system architecture**

This diagram shows the initial system architecture diagram of the L3Data RDF logging system. The FALCON user uploads a file with some CMIS information (discussed in section 6.3) using the L3Data RESTful API (see 6.4). This file is then placed in a folder on the Alfresco CMS. Once the XLIFF file is in the folder it will be logged by the provenance logger (described in section 6.5).

## Initial L3Data Functionality

The three main functions of the L3Data API are as follows:

* Registering L3Data Components
* Provenance Generation
* Provenance Data Retrieval



**Figure 5; L3Data bootstrap architecture**

### Registering L3Data Components

A component must register with the service before it can avail of the provenance generation or data retrieval services. The **POST** for registering a L3Data component using the L3Data REST API looks like the following:

***http://phaedrus.scss.tcd.ie/gls/L3DataLog/register/?componentName=(...)&password=(...)&partitionName=(...)***

This **POST** function for registering via the L3Data REST API is wrapped up in the L3Data Bootstrap API in the **alfresco.L3Data** package in the **L3Data** class. The function is as follows:

**registerComponent*(String componentName, String password, String partitionName)***

### Adding file/files to Alfresco

To push content to Alfresco via the L3Data bootstrap API the user has to use the **alfresco.L3Data** and the **L3DataBootstrap** class. This class will then run the following function which is located in the **alfresco.repo** package in the **RepositoryHandler** class and is as follows (or they can run the function directly):

**sendContentToCMS*(String filename, String activity, String user, String password, String content, String contentType, String component)***

The user will have to indicate which file or files they wish to upload, the content type or types, the activity or activities , their user name, password and component name which they gave when they registered **(future work: *registration information could be automatically detected via a configuration file)***.

The access to which Alfresco folders users are allowed to write to is determined via their user name and password which will be located in a configuration file located on their machine. When the content is being uploaded to Alfresco if the file doesn’t exist on the machine then a new file is created and the CMIS L3Data properties are added to file. Otherwise if the file already exists then the file is checked out and then updated and then checked back into Alfresco and the CMIS L3Data properties are updated.

### Provenance Generation

Once the file is uploaded to Alfresco the **L3Data:processed** CMIS object is marked as ***“no”*** and this means that the file has not been processed for provenance logging. The next step in the process is creating the provenance log via the L3Data REST API and setting the CMIS object L3Data**:processed** to ***“yes”*** indicating the file has been successfully processed for provenance logging.

To post content to the Jena triple store we use the L3Data REST API and then use the provenance generation service. The **POST** for the L3Data REST API looks like the following:

***http://phaedrus.scss.tcd.ie/gls/L3DataLog/service/(your\_component\_name)/?password=(...)&activity=(...)&user=(...)&contentConsumed1=(...)&contentConsumed1Type=(...)&contentGenerated1=(...)&contentGenerated1Type=(...)***

This **POST** function for generating provenance via the L3Data REST API is wrapped up in the L3Data Bootstrap API in the **alfresco.L3Data** package in the **L3Data** class. The function is as follows:

**postL3DataContent*(String componentName, String user, String password, String activity, String contentConsumed, String contentConsumedType,String contentGenrated, String contentGeneratedType)***

This function will return a string. If the string is blank the provenance generation has been successful and there have been no errors. This means that the CMIS object L3Data**:processed**  is set to **“yes”** indicating the provenance has been generated successfully. Otherwise, if the string is not blank then there has been an error with the post and the CMIS object L3Data**:processed** stays set to **“no”.** This method of error handling helps to catch any files that have not processed via L3Data’s provenance generation service.

## Initial CMIS Model

### What is a CMIS object?

Content Management Interoperability Services (CMIS) is an open standard that allows different content management systems to inter-operate over the Internet. CMIS defines an abstraction layer for controlling diverse document management systems and repositories using web protocols.

CMIS provides a common data model covering typed files and folders with generic properties that can be set or read. There is a set of services for adding and retrieving documents ('objects'). There may be an access control system, a checkout and version control facility, and the ability to define generic relations.

### L3Data CMIS Model

The L3Data custom CMIS model can be seen in the Git Hub at the following address: [https://github.com/CNGL-repo/IA/blob/master/L3DataBootstrap/L3DataModel.xml](https://github.com/CNGL-repo/IA/blob/master/GlobicBootstrap/globicModel.xml)

The model is installed in Alfresco in the **WEB-INF/classes/extensions** folder where simply the **L3DataModel.xml** is just placed.

This model is used to verify which properties can be associated with a file and what type these properties can be *i.e. text, int, Boolean etc*. Every time a new file is added to Alfresco using the L3Data bootstrap API the following CMIS object properties are associated with the newly created file *(as described by the CMIS model in* ***L3DataModel.xml****)*.

The six L3Data CMIS properties are as follows:

* **L3Data:user -** The name of the user who added or updated the file in Alfresco.
* **L3Data:password -** Every component in L3Data has a password ***(given at component registration)***. If this is incorrect the file cannot be processed for provenance generation by the L3Data REST API.
* **L3Data:activity** - This is the activity that is being carried out by your component. An activity is limited to the list of content processing activities specified in the L3Data semantic model. In order for the service to be able to recognise the activity, the name you give to the activity being carried out must contain the actual name of the activity as specified in the L3Data model.
* **L3Data:content –** This describes whether the file has been **“consumed”** by an activity or **“generated”** by an activity.
* **L3Data:contentType -** This is an optional CMIS property if it is not set it will use a default value. This is used to describe what type is being used for consumption or generation of an activity based on the L3Data model. More details can be found here [http://kdeg-vm-24.scss.tcd.ie/L3DataLog/user\_guide.html](http://kdeg-vm-24.scss.tcd.ie/globicLog/user_guide.html).
* **L3Data:processed -** this indicates whether a file has been processed for provenance generation correctly or not. If **L3Data:processed** is **“yes”** then provenance generation has been successful. Otherwise, if **L3Data:processed** is ***“no”*** then provenance generation has not been successful ***(this indicates an error has taken place)***.

## Initial RESTful service

### Registering L3Data Components

A component must register with the service before it can avail of the provenance generation or data retrieval services.  
Registration with the system is carried out via a URI request and uses the **POST** method. The structure of the request is important for the service to recognise what needs to be carried out.

**Sample URL:**

PUT **http:// *phaedrus.scss.tcd.ie*/L3DataBootstrap/registerL3DataComponent**

**Media-Type:** text/plain

**Values:**

* + **&componentName**: Here you specify what name you want to call your component. Component name must be unique. It is not possible to have two or more components with the same name registered with the service.
  + **&password**: Here you specify your password which will be used to verify your component when making requests to the system. It is recommended not use a password you have for any other accounts.
  + **&partitionName**(optional) : Here you can specify a partition in the triple-store in which your component will store data to and retrieve data from. This is an optional token in the request. If it is not specified, you will record data to and retrieve data from a default partition in the triple-store. This option can be useful if you want your own 'sandbox' so provenance records from other components will not be appear in your partition. It is also possible for multiple components to specify the partition name if they wish to share the same provenance data.

**Output: TBC**

### Provenance Generation

This describes how to submit data to the service in order for it to generate provenance data in RDF making special use of the [L3Data semantic model](http://www.cngl.ie/ontologies/gic) and the [PROV-O semantic model](http://www.w3.org/ns/prov). A component must be registered with the service before it can submit data. Submitting data to the service is done in a similar manner to registration, through a URI request and use of the **POST** method. Like the registration request, the structure of the provenance generation request is important for the service to recognise what needs to be carried out.

**Sample URL:**

POST **http://phaedrus.scss.tcd.ie/L3DataBootstrap/postL3DataData**

**Media-Type:** text/plain

**Values:**

* + **password** : Here you enter your password.
  + **componentName**: Here you specify what name you want to call your component. Component name must be unique. It is not possible to have two or more components with the same name registered with the service
  + **activity** : Here you enter the activity that is being carried out by your component. An activity is limited to the list of content processing activities specified in the L3Data semantic model. In order for the service to be able to recognise the activity, the name you give to the activity being carried out must contain the actual name of the activity as specified in the L3Data model.  
    For example, two content processing activities within the L3Data model are *TextAnalysis* and *Translate*. Some Sample activity names that would work with the service would be : textanalysis\_50, 5\_My\_TextAnalysis, translateActivity101 or my\_translate\_92. These work as the actual activity name appears within the name given. Some Sample activity names that would not work with the service would be: 5\_My\_Text\_Analysis, AnalysisText\_12 or trans\_7.  
    A list of the content processing activities from the L3Data semantic model and the types of content that an activity can consume or generate can be found [here](http://www.scss.tcd.ie/~meehanal/globicLog/activity_content_list.html).
  + **user** (Optional) : Here you can specify if a user was involved in an activity along with a component. This is an optional token in the request as a user does not always have to be involved in an activity. The user can be the name of a person or a worker ID number etc. If specifying the user as the name of a person with a first and last name, do not use a space between the names, instead use humpback notation or use an underscore, for example: JoeBloggs or Joe\_Bloggs.
  + **contentConsumed#** (Optional \*) : Here it is possible to specify up to 10 pieces of content that were consumed or used by an activity. This is an optional token in the request as not all activities consume content. Pieces of content consumed are specified by appending a number (1-10) at the end of the token, where the hash symbol is located. The L3Data Log Service deals with content from a file oriented perspective, so what you specify is a URI of the files location.  
    An example to specify two pieces of consumed content, *document\_1* and *document\_2* with a URI location of *http://www.example.com/filestore/*, would look as follows: &contentConsumed1=<http://www.example.com/filestore/document\_1>&contentConsumed2=<http://www.example.com/filestore/docuemnt\_2> .
  + **contentConsumed#Type** (Optional) : Here it is possible to specify the *type* of any of the consumed content. The *type* describes what a piece of content is i.e. a string, a word, a sentence, a structure, a translation, a revise translation, a piece of analyised text etc. The L3Data semantic model contains the types of content that can be consumed for each activity so when specifying a type for a piece of consumed content, it must match that in the L3Data model.This is an optional token in the request; it is used to make the generated provenance data more accurate. If it is not set then any consumed content is set to a default type. In order to specify a type for a piece of consumed content, the number associated with a piece of consumed content is placed in the token, where the hash symbol is located.  
    For example, if you have already specified some consumed content: &contentConsumed1= . In order to declare that document\_1 is of type *nif:String*, you would add the following to the request: &contentConsumed1Type=nif:String .  
    The content types must use the appropriate namespaces. A list of the content processing activities from the L3Data semantic model and the types of content that an activity can consume or generate can be found [here](http://www.scss.tcd.ie/~meehanal/globicLog/activity_content_list.html).
  + **contentGenerated#** (Optional \*) : Here it is possible to specify up to 10 pieces of content that were generated by an activity. This is an optional token in the request as not all activities generate content. Pieces of content are specified by appending a number (1-10) at the end of the token, where the hash symbol is located. The L3Data Log Service deals with content from a file oriented perspective, so what you specify is a URI of the files location.  
    An example to specify two pieces of content that were generated, *document\_3* and *document\_4* with a URI location of *http://www.example.com/filestore/*, would look as follows: &contentGenerated1=<http://www.example.com/filestore/document\_3>&contentGenerated2=<http://www.example.com/filestore/docuemnt\_4> .
  + **contentGenerated#Type** (Optional) : Here it is possible to specify the *type* of any of the generated content similar to how you can do it for any content consumed. The L3Data semantic model contains the types of content that can be generated for each activity so when specifying a type for a piece of generated content, it must match that in the L3Data model. This is an optional token in the request; it is used to make the generated provenance data more accurate. If it is not set then any consumed content is set to a default type. In order to specify a type for a piece of generated content, the number associated with a piece of generated content is placed in the token, where the hash symbol is located.  
    For example, if you have already specified some generated content: &contentGenereated1=<http://www.example.com/filestore/document\_3> . In order to declare that document\_3 is of type *gic:Translation*, you would add the following to the request: &contentGenerated1Type=gic:Translation .  
    The content types must use the appropriate namespaces. A list of the content processing activities from the L3Data semantic model and the types of content that an activity can consume or generate can be found [here](http://www.scss.tcd.ie/~meehanal/globicLog/activity_content_list.html).

(**Optional \*** = While the '&contentConsumed#' and '&contentGenerated#' tokens are optional themselves, at least one must appear in a request as an activity must consume or generate at least one piece of content.)

**Output: TBC**

### Provenance Data Retrieval

This describes how to retrieve provenance data from the service about a piece of *content*, a *component*, an *activity* or a *user*. The data is returned in JSON format. A component must be registered with the service before it can retrieve data. Retrieving data from the service is carried out via a URI request and uses the **GET** method. The structure of the data retrieval request is important for the service to recognise what needs to be carried out.

**GET http://phaedrus.scss.tcd.ie/L3DataBootstrap/retrieveL3DataData**

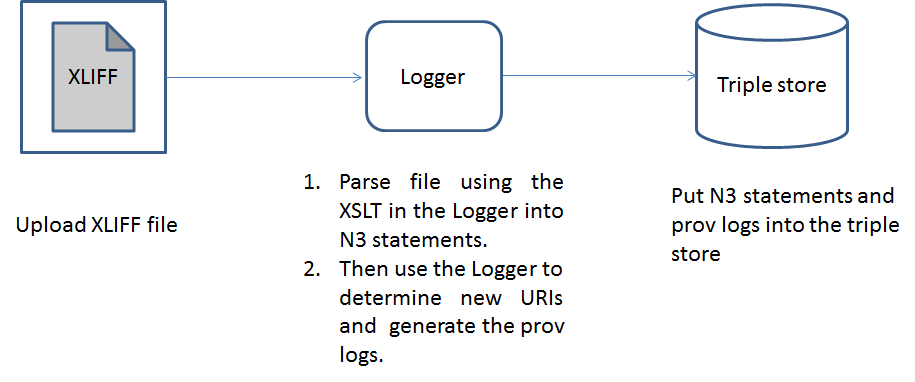
**Media-Type:** application/json

**Values:**

* + **password**: Enter your L3Data password.
  + **componentName**: Here you specify what name you want to call your component. Component name must be unique. It is not possible to have two or more components with the same name registered with the service
  + **retrieveData**: Here it is possible to specify the name of some data for which you wish to retrieve provenance records about. It is possible to retrieve provenance records about a piece of *content*, a *component*, an *activity* or a *user*. For example:  
    To retrieve provenance records about a piece content, you would add the following to the request: &retrieveData=<http://www.example.com/filestore/document\_1>.  
    To retrieve provenance records about a component, an activity or a user, you would add the following to the request:   
    **Component:** &retrieveData=gic:MockComponent\_1   
    **Activity:** &retrieveData=gic:translateActivity101   
    **User:** &retrieveData=gic:Joe\_Bloggs

**Output**: TBC

## Initial Provenance Logging



**Figure 6: Initial provenance logging system**

### Prov Logger Overview

The Logger component is a servlet which receives successive versions of XLIFF files, transforms this XLIFF input into RDF statements and merges these statements into the triple store. It also generates extra RDF statements to track provenance information.

### Initial Provenance Logging Overview

Logger performs the following tasks for each input file it receives:

1. Transform the input XLIFF file into a set of RDF statements. This is done through an XSL transformation which processes the XLIFF document to produce N3 statements.
2. Merge these RDF statements into the triple store.
3. Add provenance information. A new "log" URI is created with associated statements, such as a timestamp and an input "job", and is linked using a provenance statement to each of the newly added URIs that have resulted from the merge.

### Initial Provenance logging algorithm

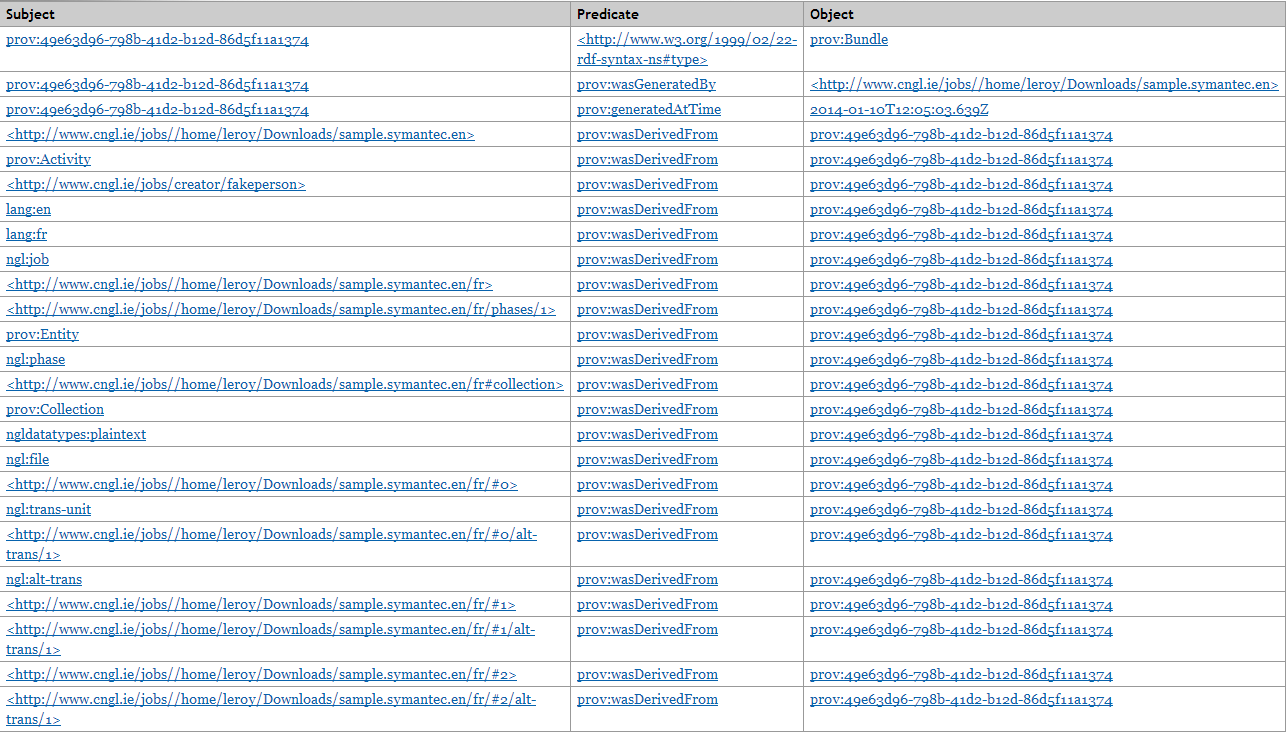
1. Upload a XLIFF document.
2. Parse the XLIFF document using XSLT into N3 statements.
3. Then check the generated N3 statements URIs against the URIs in the triple store. This check is done at the merge stage where new URIs and data are added to the graph and old URIs are being merged with the graph along with old/new data
4. A prov id is generated by a UUID generator which is available **java.util.UUID** package. A UUID is required as the provenance URI has to be unique. Then do the following dependent on the results of step 3:

*if(there are new URIs) then*

*create a new prov record which indicates what file has generated this new prov record, the time the prov record is generated at and the new URIs that were added into the triple store*

*else then*

*just make a prov record which record which indicates what file has generated this prov record and the time the prov record is generated at*



**Figure 7: Sample provenance log**

### Updating the XLIFF to RDF transformation

The transformation of XLIFF files to RDF is done through the XSLT document in Logger/src/xliff2n3/xliff2n3.xsl. All statements other than the provenance statements (which are added programmatically in Logger/src/xliff2n3/Graph.java using URIs defined in Logger/src/xliff2n3/Constants.java) originate from this XSLT document, which can be tweaked to replace the grammar and the URIs used. Namespaces are included in this XSLT document too, and will be automatically added to the triple store server namespace list. After making any changes to the XSLT, you will probably want to clear the triple store and re-process the sample files to see the effect of your changes.

### Initial schema for resource identification

The process of logging iteratively into the triple store relies on the use of unique identifiers, which make it possible to determine when statements found in successive input files refer to identical resources. This is a challenge for XLIFF documents, which do not provide intrinsic strong identifiers. For instance, the job-id tag is optional (as per the OASIS XLIFF schema) and applies to each file element; there is no equivalent document-level. The XSLT constructs URIs based on the job id (assuming it is present and unique, which is true in the sample files but may not hold in some cases) to which names are appended in a manner that matches the structure of the XLF document e.g. <http://www.cngl.ie/jobs/b043a10ead/content.txt/#2/MT-Leverage/google>. This can of course be tweaked by changing the XSLT document. If necessary, an external parameter (a unique id from another system for example) can be passed to XSLT processor as an input parameter and used in the XSLT document.

### Technologies used

#### Triple Store

Lastly, an off-the-shelf triple store and it associated web service provides data persistence and allows the other components to add triples to the store and retrieve triples through SPARQL queries. The application should work with any SPARQL endpoint (note: has been tested with Sesame[[25]](#footnote-25) as well as Jena).

#### XSLT

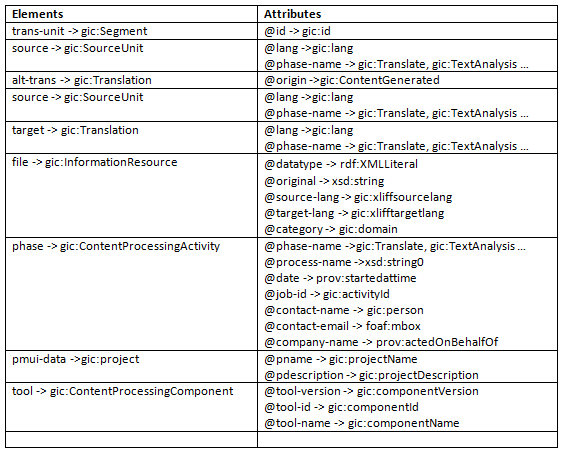
The conversion of XLIFF files to RDF statements is done using XSLT. This makes it easy to modify and tweak the conversion process outside of the application code. We use the N3 format as the XSLT conversion output as it syntactically simple and easy to read. XSLT conversion is followed by post-processing to add provenance information.

#### Optional XLIFF validation

Logger optionally validates input against the XLIFF schema. If validation is enabled and XLIFF input is found to be non-conformant, processing will be aborted with the relevant validation error message in the response. Note that this feature requires Internet connectivity as the OASIS XLIFF schema references external elements on the WWW, and proxy settings may therefore need to be configured in Logger (in file config.properties) if XLIFF validation functionality is enabled.

## Initial L3Data schema mapping

The initial mapping schema for the XLIFF mapping to RDF will be an ontology developed in CNGL called GLOBIC. The mapping to GLOBIC from a vanilla XLIFF file can be seen below:



**Table 1: summary of mapping from XLIFF nodes to L3Data**

The collation and addition of meta-data terminology is another important aspect of the project. For adding this meta-data to the XLIFF document we are going to use the W3C standard Internationalization Tag Set 2.0 (ITS 2.0). ITS 2.0 is a set of attributes and elements designed to provide [internationalization and localization](http://en.wikipedia.org/wiki/Internationalization_and_localization) support in XML and HTML5 documents. The standard for mapping ITS 2.0 to XLIFF can be found at the following address <http://www.w3.org/International/its/wiki/XLIFF_Mapping>. Then for mapping XLIFF+ITS to RDF we will use the mappings found at this address <http://www.w3.org/International/its/wiki/ITS-RDF_mapping>.

1. Conclusions and Next Steps

This document provides an initial reference architecture for the integration and use of linguistic linked data into localisation workflows using the L3Data Schema. An initial evaluation of this architecture will be performed in 2014 though integration of the L3Data Reference Implementation with a commercial localisation tool chain consisting of XTM Cloud, TermWeb and EasyLing. This will provide feedback on the integration and performance capabilities of the integrated system as well as demonstrating the added value provided in terms of end-to-end process quality monitoring and the retraining of MT and term extraction components to improve their performance.

Based on this feedback the L3Data Architecture and Schema will be revised and subjected to another implementation and evaluation cycle in 2015. In addition however, there are several activities that being conducted in parallel to FALCON or that offer more specific functionality now addressed above, and these will also be examined revising the L3Data platform into 2015. These activities include:

* Language Resource Sharing using Linked Data: The language technology research community in Europe has a well-established XML schema for language resource meta-data and a sharing network for this meta-data established by the META-SHARE project[[26]](#footnote-26). However, this community is exploring migration of this meta-data to a linked-data infrastructure.
* The XLIFF technical committee are aiming to complete version 2.0 of this standard by 2015[[27]](#footnote-27). This includes the ability to add several modules capturing further meta-data, including an ITS2.0 module.
* The OntoLex Community Group[[28]](#footnote-28) at the W3C is finalising a specification for integrated lexical and ontological information. This is developed from the LEMON ontology for sharing lexical knowledge and is similar to the model used for the BabelNet lexical/encyclopaedic linked data resource[[29]](#footnote-29). If adopted more widely this would form a strong target schema for linked into open lexical resources and therefore used in annotations by third party text annotation services.
* The Best Practices for Multilingual Linked Open Data Community Group (BP-MLOD) [[30]](#footnote-30) at the W3C is defining best practice guidelines relevant to the sharing of multilingual resources as linked data. This will be followed closely and its recommendations incorporated into the revised architecture and schema.
* The Linked Data for Language Resources (LD4LT)[[31]](#footnote-31) Community Group is gathering industrial use cases and requirements for the use of linguistic linked data, in particular with language technologies such as content analytics and machine translation. FALCON will both contribute and leverage these activities as part of the development of this architecture.

1. http://falcon-project.eu/wp-content/uploads/2013/09/FALCON-D2.1-submit.pdf [↑](#footnote-ref-1)
2. http://www.w3.org/TR/prov-o/ [↑](#footnote-ref-2)
3. http://www.w3.org/2005/11/its/rdf-content/its-rdf.rdf [↑](#footnote-ref-3)
4. http://docs.oasis-open.org/xliff/v1.2/cs02/xliff-core.html [↑](#footnote-ref-4)
5. https://code.google.com/p/m4loc/ [↑](#footnote-ref-5)
6. http://www.w3.org/International/multilingualweb/lt/ [↑](#footnote-ref-6)
7. http://www.w3.org/International/its/wiki/XLIFF\_1.2\_Mapping [↑](#footnote-ref-7)
8. http://kilgray.com/memoq/2013/help-en/index.html?xliffdoc.html [↑](#footnote-ref-8)
9. https://code.google.com/p/interoperability-now/ [↑](#footnote-ref-9)
10. https://code.google.com/p/interoperability-now/ [↑](#footnote-ref-10)
11. http://www.gala-global.org/oscarStandards/tmx/tmx14b.html [↑](#footnote-ref-11)
12. http://www.ttt.org/oscarStandards/tbx/tbx\_oscar.pdf [↑](#footnote-ref-12)
13. http://www.linport.org/terms.php [↑](#footnote-ref-13)
14. http://wwww.ttt.org/specs/ [↑](#footnote-ref-14)
15. https://interoperability-now.googlecode.com/files/The\_TMS\_Interoperability\_Protocol\_Package-1.5.pdf [↑](#footnote-ref-15)
16. http://www.cngl.ie [↑](#footnote-ref-16)
17. http://nlp2rdf.org/ [↑](#footnote-ref-17)
18. http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html#d4e301 [↑](#footnote-ref-18)
19. http://www.w3.org/TR/its20/#conversion-to-nif [↑](#footnote-ref-19)
20. http://www.xtm-intl.com/xtmcloud [↑](#footnote-ref-20)
21. http://www.interverbumtech.com/ProductsServices/TermWeb.aspx [↑](#footnote-ref-21)
22. http://www.easyling.com/website-translators-agencies/ [↑](#footnote-ref-22)
23. https://www.oasis-open.org/committees/tc\_home.php?wg\_abbrev=cmis [↑](#footnote-ref-23)
24. http://www.w3.org/TR/sparql11-query/ [↑](#footnote-ref-24)
25. http://www.openrdf.org/ [↑](#footnote-ref-25)
26. http://www.meta-share.eu/ [↑](#footnote-ref-26)
27. https://wiki.oasis-open.org/xliff/XLIFF2.0 [↑](#footnote-ref-27)
28. http://www.w3.org/community/ontolex/ [↑](#footnote-ref-28)
29. http://babelnet.org/ [↑](#footnote-ref-29)
30. http://www.w3.org/community/bpmlod/ [↑](#footnote-ref-30)
31. http://www.w3.org/community/ld4lt/ [↑](#footnote-ref-31)