
Kevin Bargary


Peter Reynolds

Using Web Services for Translation

By Kevin Bargary and Peter Reynolds

Abstract

Web services use Internet technologies to allow computer-based systems to communicate and transfer data in a way that provides a more seamless automated workflow. At OASIS work is being done to create a standard way for Web services to be used within the translation and localisation industry. The purpose of this article is to inform you about this work, what Web services are, and to outline a real-life case study showing how this technology is being put to use now. The article will deal in detail with the specification proposed by the OASIS technical committee for Translation Web Services. It will also describe use cases where this specification can be put into practise such as the project implemented by the Localisation Research Centre as part of the IGNITE project.

Keywords

Open standards, Web services, translation, localisation life cycle, standards development, OASIS, localisation, localization, globalization, globalisation, Localisation Research Centre, LRC

1. Introduction

Web services is a solution to the problem of computer systems not talking to each other. It uses Internet technologies to allow computer-based systems to communicate and transfer data in a way that provides a more seamless automated workflow. The likelihood is that Web services will be adopted by companies over the next few years to automate processes and integrate systems. Within the translation and localisation industry, however, there is work being done to create a standard way for Web services to be used. This work is being done at OASIS, which is the organisation for creating standards, particularly XML standards within the industry. The purpose of this article is to inform you about this work, what Web services is, and to outline a real-life case study showing how this technology is being put to use now.

The idea of creating a standard for the use of Web services within translation was put forward by Bill Looby of IBM at the eLocalisation 2001 conference held in Limerick, Ireland. Mr. Looby delivered a paper which showed a vision of how the industry could benefit from agreeing on a common way of using this technology. The conference had also seen a practical demonstration of how this technology was already being used. Lionbridge (then Berlitz GlobalNET) gave a demonstration of work being done for the 2003 Special Olympics Web site, which it was sponsoring. Using Web services, an XLIFF file was sent from the Web site to Elcano, Lionbridge's online translation service, and back to the Web site.

This conference ended with a small group of people getting together to look at how they could progress with the idea of using Web services within the translation industry. The steering group that formed decided that OASIS would be the natural home. OASIS was established in 1993 and it is focussed on XML standards for the software industry. XLIFF (XML Localization Interchange File Format) was already being developed by an OASIS technical committee, and there was considerable support for this industry within OASIS. The OASIS technical committee was formed at the beginning of 2003 and its members include representatives from Oracle, Microsoft, IBM, Connect Global Solutions, thebigword, LISA, the LRC, and Lionbridge as well as individual members.

2. Translation Web Services

2.1 What are Web Services?

Before detailing the main features in the draft specification from the Translation Web Services (TWS) technical committee, we would like to give some background on Web services. The World Wide Web is a collection of interlinked documents that sits on the Internet, which is effectively a huge computer network that connects individual Web sites. To access a Web site a person sits at a computer and views pages through a Web browser – a process that can be considered as machine-to-person communication. With the advent of Web services the Internet is used for machine-to-

machine communication rather than machine-to-person communication. Protocols such as HTTP and standards such as XML and SOAP (Simple Object Access Protocol) are used in Web services to enable this machine-to-machine communication. This allows different systems to work together, allowing for more powerful functionality and automation.

A Web service might do something such as enable weather forecasts to be queried by remote computers over the Internet. To do this the weather forecasting company would have to create a Web service and allow it to be accessed. This is done using an XML document called a Web Service Definition Language (WSDL) document. The WSDL document describes the services which the client application is allowed access to and describes what parameters will be sent and received for each of these calls. A gardening enthusiast who is away a lot might want his computer to control his water sprinkler. By using Web services the computer will be able to find out when it is not raining and turn on the sprinkler. A protocol called Simple Object Access Protocol (SOAP) is used for this.

2.2 What is Translation Web Services?

Since January 2003 the Translation Web Services (TWS) technical committee has been working to create a standard way for Web services to be used in a multilingual context. It has concentrated its efforts on creating a standard relating to the communication between publisher and vendor companies. At the simplest level this will allow for translation and other work to be sent by the publisher to the vendor and, once translated, sent back. The draft specification covers the following areas:

- Service support
- Translation and request quote
- Status, notification and delivery
- Reference files
- Security

3. Translation Web Services Specification

Each service in the TWS specification provides two forms for interaction between client and vendor. These forms are *request* and *response*. For example, the client submits a *retrieveServiceList* request to the vendor. The vendor receives this request, processes it, and then returns a *retrieveServiceList* response to the client. The *request* and *response* forms of a service expect different inputs and produce different outputs but both *request* and *response* are needed for the interaction between and use of each service.

3.1 Categories of Services

The TWS specification defines five categories of methods or services, namely 'Service Support', 'Security', 'Translation & Request Quote', 'Status, Notification and Delivery' and 'Reference Files'. These categories form a guideline for the services that the TWS specification provides. Each category encapsulates one facet of the core work required for the completion of a translation job, from initial quote through to final delivery.

3.1.1 Service Support

The 'Service Support' category contains only one service, namely *retrieveServiceList*. This service allows the client to query the vendor on the type of localisation services they provide. When a *retrieveServiceList* request is made on a vendor a list of languages, service types, domain types, and MIME (Multipurpose Internet Mail Extensions) types that are supported is returned. In the case where no relationship exists between client and vendor, the *retrieveServiceList* is the first service evoked by the client to ensure the vendor meets the requirements for the potential translation job. The client can then use the information returned in their future interactions with the vendor.

3.1.2 Security

As with any transaction over the web, data security is an important consideration. OASIS defines a Web services security standard specification (WS-Security) which provides several methods for the securing of Web service-related transactions. The TWS specification relies on WS-Security to provide an end-to-end message level security and hence the specification recommends the use of username/password-based security over SSL.

3.1.3 Translation and Request Quote

This category details the services required to instantiate a job between a client and a vendor. Web services for translation uses a job ticket as a unique identifier for each project. This job ticket is created on the client side usually before a quote request. The job ticket consists of a project ID, a user ID and a unique job ID. This job ticket can then be used in all future interactions with the vendor's web service. Currently there are two methods of initiating a job in the 'Translation and Request Quote' category.

The first method is where the client submits a *requestQuote* service. The *requestQuote* service details the information pertaining to the translation job (word count, languages, etc.). The client retrieves the generated quote using the service *retrieveQuote* and chooses to accept or reject the quote. If the quote is accepted an *acceptQuote* service is activated; if it is not accepted the generated quote will expire after a certain time limit, defined by the vendor.

The second method is based upon the *submitJob* service. The submitJob service has similar inputs to the *requestQuote* service but it also contains the purchase order information found in the *acceptQuote* service used in the previous method. In using this second method it is automatically assumed that the job will be accepted. This interaction might be between two in-house systems, e.g. one system has content to be translated, and it contacts a second MT system and gets the content translated.

3.1.4 Status, Notification and Delivery

The Translation Web Services technical committee provides seven status, notification and delivery management services

in the TWS specification. This set of services allows the client some control over the work that is being carried out by the vendor for a particular job. Using these services a client can check the status of and cancel or suspend a particular job. The success of each service request is dependant on the status of the job at the time of calling the service. For example you cannot cancel a job that has already been completed (for obvious reasons).

The *retrieveActiveJobsList* service returns to the client a list of all active jobs that they have with a particular vendor. An alternative to this is the *retrieveFullJobsList* service, which returns all jobs associated with a particular vendor irrespective of the current status.

A client can query the vendor using the *retrieveJobInformation* service and get a response containing all current information about a job. The status of the job can be deduced from the information received from the *retrieveJobInformation* service and possible changes to the project deadlines can be predicted. If a job is completed then the status of the *retrieveJobInformation* service response should reflect this.

If the information received back from the vendor after a *retrieveJobInformation* service request indicates that the job is completed, this job can then be downloaded using the *retrieveJob* service.

The client can choose to suspend a job temporarily at any time as long as the job status is not complete. This is done by making a *suspendJob* service request.

To remove this temporary suspension of a job (by submitting a *suspendJob* service request), the client can choose to resume the job using the *resumeJob* service.

A client can cancel a job using the *cancelJob* service if the job is currently active and not in a completed state.

3.1.5 Reference Files

The vast majority of localisation projects require not only the localisable content but also any reference files associated with the project. Reference files are not for translation but contain information that may help the translation process. Translation memories, style guides, or terminology references may be sent along with the translatable files. The 'Reference' category defines services to allow for this allocation of these files to a particular project.

A resource file can be assigned to any number of active jobs using the *associateResource* service.

To remove an association between a job and a resource file the *disassociateResource* service is used.

The client can review information about a resource file by invoking the *retrieveResourceInformation* service. This will return information about the resource file from the vendor: a list of jobs that the resource file is assigned to, the purpose of the resource file and whether or not the file has changed (been updated).
The TWS specification allows the client the functionality of uploading assets to the vendor using SOAP messages using the *uploadFile* service.

3.2 Services Supported in Current Specification

At this point there are 18 services supported by the TWS specification. As discussed in section 3.1, services can be categorised into one of five categories depending on their function in a translation process. These services can conversely be considered under the following three headings:

Required Services – these are services that are required for a Translation Web Services implementation and form the basis of a minimalist approach to Translation Web Services use.

Recommended Services – these services are recommended by the Translation Web Services technical committee to be used in an implementation of Translation Web Services together with the 'required' services.

Optional Services – these services are services that are only needed in some specific cases (enquiring about what services a vendor has to offer or setting up a first contact with a vendor by requesting a quote etc.). These services are not essential to the Translation Web Services process but are required for some scenarios.

Required Services	Optional Services	Recommended Services
submitJob	retrieveServiceList	rejectJob
retrieveJobInformation	requestQuote	associateResource
retrieveJob	acceptQuote	disassociateResource
retrieveActiveJobsList	retrieveQuote	retrieveResourceInformation
suspendJob	retrieveFullJobsList	retrieveFullResourceList
resumeJob		uploadFile
cancelJob		

Table 1: *List of services available in the TWS specification*

4. Technologies in Translation Web Services

4.1 Simple Object Access Protocol (SOAP)

SOAP is a W3C-developed standard described as a communication protocol or a message passing system between two computers. SOAP is the specification that defines the XML format for these messages being passed. SOAP is one of three core XML-based standards that are the foundation of a Web services implementation (the others being WSDL and UDDI, see Sections 4.2 and 4.3).

4.2 Web Services Description Language (WSDL)

A WSDL is an XML document that describes (a) a set of SOAP messages and (b) how these messages are exchanged. The WSDL file in a practical sense contains the information required by a client to access a service, i.e. what parameters

5. Localisation Life Cycle

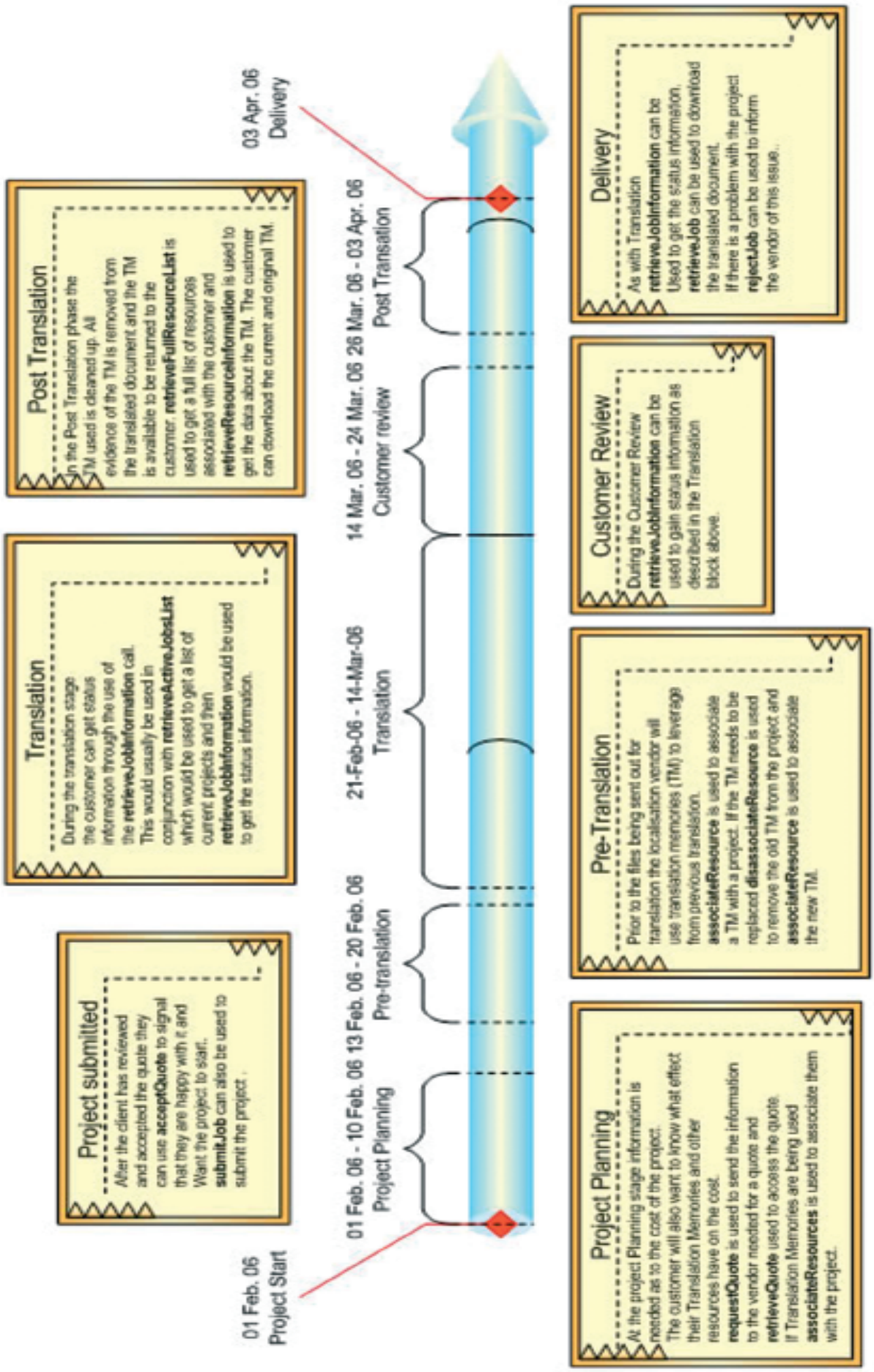

Figure 1: *Localisation life cycle incorporating the use of Web services for translation*

need to be passed to the service to invoke a response. The WSDL file should also contain the actual location (HTTP address) of the web service.

4.3 Universal Description, Discovery and Integration (UDDI)

UDDI is an OASIS-driven mechanism for clients to dynamically find other Web services. The UDDI protocol gives a company the ability to register their available Web services online, thus exposing them to potential clients. An UDDI registry service is a Web service that manages information about service providers, service implementations, and service metadata.

In summation, SOAP is the communication protocol for Web services, WSDL defines how the interaction occurs between the two computers, i.e. how to invoke the services and the UDDI is a mechanism for finding these services or registering one's own services.

5. Localisation Life Cycle

See page 10, Figure 1: Localisation life cycle incorporating the use of Web services for translation.

6. Use Cases

6.1 TWS Reference Implementation

A reference implementation of the Translation Web Services specification was undertaken by the Localisation Research Centre at the University of Limerick in Ireland as part of the IGNITE project. The Translation Web Services technical committee decided that it was important to have a reference implementation to see how the standard worked from a technical viewpoint.

The basic premise of Web services for translation is that a server machine will contain a pre-programmed set of methods or functions that a client machine can access using Web services technology. The two components involved in this interaction are the server machine and the client machine. The process of implementing Web services for each component is similar. Nevertheless there are some subtle but important distinctions to be made between both implementations.

The implementation platform of choice for this reference implementation is J2EE and the Java programming language. The rationale behind this decision was that the open source 'Apache Project' has a Java-based implementation of SOAP called Axis. Axis is defined as a "reliable and stable base on which to implement Java Web services"; it provides an Application Programming Interface (API) into the SOAP actions that are required for implementing the Translation Web Services standard. Apache Tomcat was chosen as the Web server on which to develop the Web services. Tomcat and Axis were developed under Apache and work very well together in a practical implementation environment.

The development model used for this reference implementation was loosely based on the prototyping model of software development. Initially one of the 18 services currently available in the standard was developed. From that the development incorporated one further service at a time until all were implemented. Throughout the development life cycle we reported back to the technical committee on any issues or suggestions for improvements that arose as we progressed.

Apart from the API for the usage of SOAP functionality Axis provides two command line utilities that are essential to the implementation of Web services. The 'Java2WSDL' utility takes pre-existing Java code and creates a WSDL file for that code. This utility can be used if there is some code that performs a specific function that you would like to make available as a Web service for others to use. The Translation Web Services standard has made a WSDL available, so this utility was redundant for our purposes. However, the second utility, 'WSDL2Java', creates the Java stubs (Java files that contain the code needed to use SOAP) required by:

(a) the server to write and deploy the service, and
(b) the client to access the service through its own code.

Figures 2 and 3 show this process. Firstly the Java stubs are created:

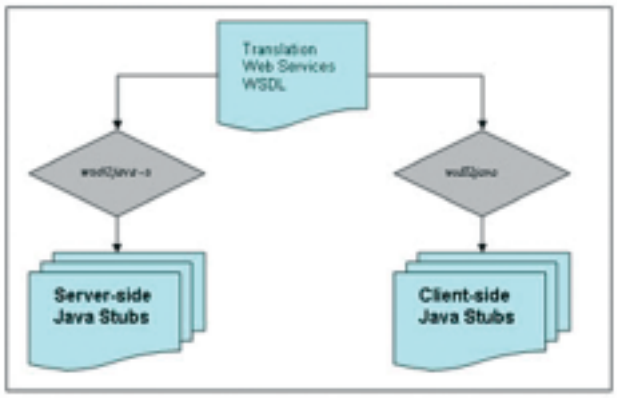

Figure 2: *Creating the Java Stubs from the WSDL*

Then the Java stubs are used by the client application to access the services and by the server to deploy the services.

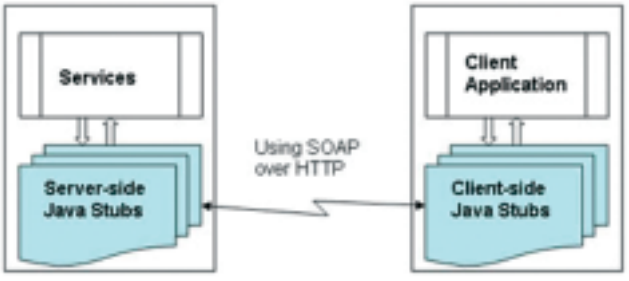

Figure 3: *Connecting client and server (through the Java stubs) over HTTP using SOAP*

The first service that we implemented was the retrieveServiceList service. This service was chosen because there were no input parameters required for it. All that was required to invoke the service was an instance of the *retrieveServiceListRequest* class. The *retrieveServiceList* service returns "a complete list of services offered by a particular vendor. This will include the languages dealt with and services offered by a particular vendor" (Translation Web Services Specification Draft 1.0). After writing the server-side code to handle a *retrieveServiceListRequest*, i.e. return all of the appropriate values, the next stage was to create a simple test class that could instantiate a *retrieveServiceListRequest* and handle the results received back from the server in a *retrieveServiceListResponse*. With both classes now ready, we needed to deploy the services to the Apache Web server. The WSDL file also contains the location of the service, i.e. where it can be accessed from. Deployment is necessary to ensure the service is in the location as defined in the WSDL.

When the utility 'WSDL2Java' creates the Java stubs needed for the server-side machine, it also creates two other files that are used to deploy and 'un-deploy' the service to a Web server (Apache Tomcat). These files are called Web Service Deployment Descriptors ('deploy.wsdd' and 'undeploy.wsdd').

The next stage in the development of the implementation was to write the code for the rest of the services. While writing the code we encountered some issues with the specification (including inconsistencies between the schema and the specification document). These issues were quickly amended by the technical committee. During the process of coding we also made some suggestions to the technical committee about possible improvements to the specification and we were actively involved in applying these changes. For example, the service *'retrieveQuote'* in the original specification did not return any information about the location of the actual quote. This was deemed to be an important piece of information for this service and was promptly included in the specification.

With the code for the implementation of the services now written, the initial service deployed *(retrieveServiceList)* was un-deployed and the full list of services was deployed to the Web server. A JSP (Java Server Pages) client interface was developed to allow for the input of the parameters required for each service and also to show the responses from the server.

6.2 Next Steps

The next step in the reference implementation will be to attach the front-end JSP interface to a back-end database system that returns some relevant information that is not pre-defined. The current implementation can be seen running live on www.electonline.org:8080/index.html. To view or download the source code used, please go to www.igniteweb.org/tws. Here you will also find instructions on how to install this implementation on your own local machine and server.

6.3 Lionbridge's use of Web services

Although the specification from the TWS technical committee is still at the draft stage, there has been some significant work done with Web services in the translation industry. Lionbridge has implemented a number of solutions based on Web services which have linked content management and other systems with Elcano, its online translation portal.

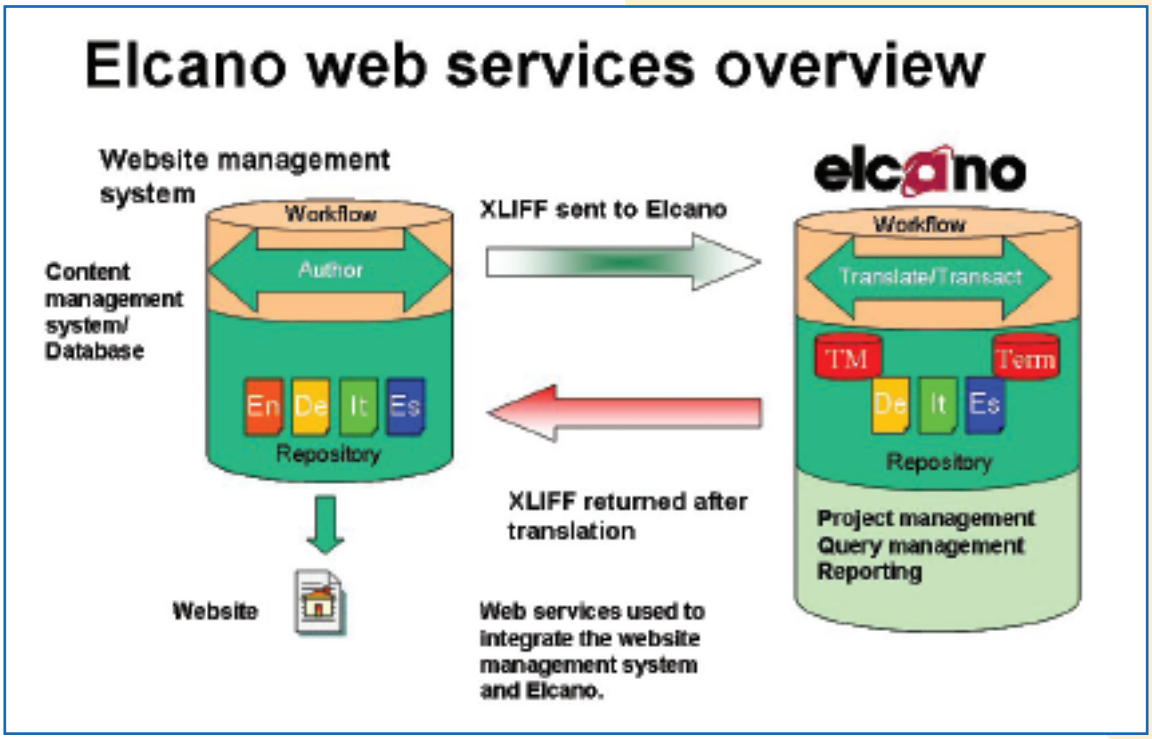

Figure 4: *Lionbridge's use of Web services*

Figure 4 shows a solution that was built for a large chemical company. Its translation process was time-consuming and error-prone. The process required more than 40 separate manual steps for each file to be translated. This led to a localisation process that was not cost-effective and delivery to target markets was unnecessarily delayed.

Working together with the company and its Systems Integrator, Lionbridge was able to propose a solution to standardise and automate parts of the process, reduce the manual effort required, and provide a more cost-effective and time-efficient way to translate the content stored within the TeamSite Content Management System (CMS).

The solution Lionbridge proposed to the company was to connect Interwoven's TeamSite to Elcano™ (For further details see http://elcano.Lionbridge.com). Elcano offers a Web services interface providing simple and direct programmatic access to Elcano from any CMS or content repository. The Elcano Web services solution makes use of two language industry standards – both under the auspices of the OASIS standards organisation (www.oasis-open.net). XLIFF defines the structure of translation data, and TWS defines the communication between TeamSite and Elcano. Using these data transfer standards to connect to Elcano allows source content of any type to be posted directly into Lionbridge's translation production process, its status to be tracked from the CMS during translation and, for completed translations to be retrieved without unnecessary manual intervention. In addition, once extracted from the CMS, Lionbridge can make use of a variety of translation productivity tools to ensure the consistency, accuracy and timeliness of the translation.

Freeway

Lionbridge recently introduced its new customer portal in April 2006 and the Elcano Web services described above will be ported to Freeway. More information is available at www.lionbridge.com.

References

Translation Web Services Technical Committee

www.oasis-open.org/committees
/tc_home.php?wg_abbrev=trans-ws

XLIFF Technical Committee

www.xliff.org

OASIS

www.oasis–open.org/
Translation Web Services and XLIFF are developed within OASIS, which is a standards body for industry.

World Wide Web Consortium

www.w3.org
The World Wide Web Consortium is the standards body for the Internet.

Localisation Research Centre

www.localisation.ie
The Localisation Research Centre (LRC) is the information, educational, and research centre for the localisation community.

IGNITE project

www.igniteweb.org
IGNITE is an LRC-coordinated project that will pool together linguistic infrastructure resources and provide convenient access and a market place for them. IGNITE hosts an implementation of the Translation Web Services specification.

LISA - OSCAR

www.lisa.org/sigs/oscar/
OSCAR is LISA's body for the development and maintenance of open standards for the language industry.

Note: If you are interested in joining the Translation Web Services technical committee please contact Peter Reynolds at peter.reynolds@lionbridge.com or Tony Jewtushenko at tony.jewtushenko@productinnovator.com

This whitepaper was written by Kevin Bargary and Peter Reynolds with additional contributions from Magnus Martikainen, Tony Jewtushenko, Andrzej Zydron and Reinhard Schaler. Kevin Bargary may be contacted at Kevin.Bargary@ul.ie and Peter Reynolds may be contacted at Peter.Reynolds@lionbridge.com