A LANGUAGE TRANSLATION SERVICE FOR DOCUMENTUM

M. Scott Roth
Director of Technology
Armedia, LL
scott.roth@armedia.com

EMC²

Overview

In our highly-connected and diverse society, the expectation that online content be available in multiple languages is greater today than ever before. Licensing agreements, rules and regulations, disclaimers, and forms on websites belonging to software vendors, credit card companies, insurers, and other service providers are expected to be available in users' native languages. For international companies, this can mean managing and maintaining these documents in multiple languages. From a content management perspective, how do you produce, manage, and maintain all of these translations? What if the "source" document changes, how do these changes ripple through to the rest of the translations? What if one of the translations needs to be "tweaked", how do you keep its version in synch with the source document?

This article discusses a solution for creating and maintaining multiple translations of content in a Documentum repository. The solution uses the inherent content management capabilities of the Documentum Content Server to manage content, versions, and relationships among documents, and leverages the Content Server's infrastructure (specifically, Service-based Objects, asynchronous jobs, and external database tables) to integrate with a translation services provider for the production of translations. The translation services provider used in this integration is Lingotek (http://www.lingotek.com). Lingotek offers a comprehensive RESTful API that integrates easily with Documentum to provide a seamless solution for the production and management of multilingual content.

Table of Contents

Disclaimer: The views, processes or methodologies published in this article are those of the author. They do not necessarily reflect EMC Corporation's views, processes or methodologies.

1 Introduction

Providing content to users in their native language is becoming the expected norm. It is not uncommon for banks, credit card companies, insurers, government agencies, and manufacturers to provide licensing agreements, rules and regulations, warranties[1], and forms[2] in multiple languages – both online and in print. These documents are usually available at an organization's brick and mortar office, or contained in their online library and are available for users to download[3]. Producing and managing documents in multiple languages can be expensive, tedious, time-consuming, and prone to staleness. For example: If the "source" document changes, how do these changes ripple through to the rest of the translations? If one of the translations needs to be "tweaked", how do you keep its version in synch with the source document? With a content management system such as EMC Documentum®, and an integration with a language translation service like Lingotek, the *process* of producing and managing multi-language content can be simplified. In this article, it is important to distinguish between internationalized or localized websites where web content and controls are provided in users' native languages[4], and documents[5] that are produced in multiple languages. This article is only concerned with the latter; that is, documents, not entire websites.

Lingotek (www.lingotek.com) is a translation services company whose translation management system lends itself well to integration with content management systems like Documentum[6]. Lingotek provides a completely online environment for submission, management, and exchange of documents for translation. They offer numerous out-of-the-box workflows to accommodate many translation needs, including machine translation, machine translation plus editor, and translation with multiple reviewers. Once a document has been submitted to Lingotek for translation, your translators can access and process the document in Lingotek's cloud-based Translation Management System (TMS). When translations are completed, they are flagged as such and returned to the originator, in their original format and style.

This article discusses an integration between the Documentum Content Server and Lingotek to provide seamless production and management of multi-language content. The integration leverages the inherent content management capabilities of the Documentum Content Server to manage content, versions, and relationships among documents, and its infrastructure to integrate with Lingotek for the seamless production of translations. The remainder of this article will discuss the design and implementation of the integration including: design goals, logic, implementation details in the Documentum infrastructure, and logging.

2 Design Overview

The design goals for this integration were simple: consolidate all interactions with the Lingotek TMS in a Service-based Object (SBO)[7] to be reused across the Documentum platform, make the translation process invisible to the user, and leverage Documentum to maintain parity among the source documents and their translations.

2.1 Use Scenario

By way of context, consider the following scenario. You are an editor for a small widget manufacturer in North America. As new widget models are produced, technical writers create assembly instructions, user's guides, and maintenance manuals for each model. All of your writers work in an editorial management system based on Documentum. Once a manual is completed, it is routed to the appropriate editor for review, editing, and final approval via a Documentum workflow. Since your website serves all of North America, you publish each manual in English, French, and Spanish regardless of the language in which it was originally written. To achieve this, you utilize the Documentum-Lingotek interface to produce the necessary translations. Once the translations are created, the document sets are published to your website for your customers to reference.

The high-level steps to fulfill this scenario, and around which the design was created, are:

1. A user selects a document for translation in the Documentum repository using Webtop. (In our scenario, this document would be attached to a workflow that is accessed from the user's Inbox.)
2. Using a customized Webtop form, the user indicates the document's source language (if it is not automatically discernable by Documentum), and the document's target language(s), and submits the document for translation.
3. The SBO transfers the document to the Lingotek TMS via the Lingotek RESTful web services API.
4. The SBO records the document's metadata in a registered table so the returning translation can be correctly associated with the original document.
5. The translation occurs in the Lingotek TMS. (There are many options for how translations occur in the TMS. In this article, I use the machine translation workflow to eliminate the need to log in to the TMS to complete the translation.)

6. Periodically, a Documentum job queries the registered table for all documents pending translations, connects to Lingotek TMS via the SBO, and tries to retrieve the translations.
7. If a translation is ready, it is imported into Documentum and related to its original document using the built-in Documentum `DM_TRANSLATION_OF` relationship.
8. The job updates the registered table and alerts the user that the translation is complete.
9. If the translation is not ready, the job flags the translation in the registered table as pending, and will try to retrieve it on its next execution.

2.2 Logical Design

The integration design is centered on the SBO, which provides all interaction with the Lingotek TMS. The SBO relies on a database table to maintain the state of each translation request and retrieval. Requests for translation are made by the user via a Webtop customization, and the retrieval of the completed translation from Lingotek is handled by an asynchronous job. Both the Webtop customization and the job leverage the SBO for all interaction with the Lingotek TMS. All other content management functions leverage the inherent capabilities of the Documentum Content Server (e.g., checkin, checkout, versioning, and relationships). The following sections discuss the logical design of the integration in greater detail.

2.2.1 Software Architecture

Figure 1 depicts the basic software architecture of the Documentum – Lingotek integration.

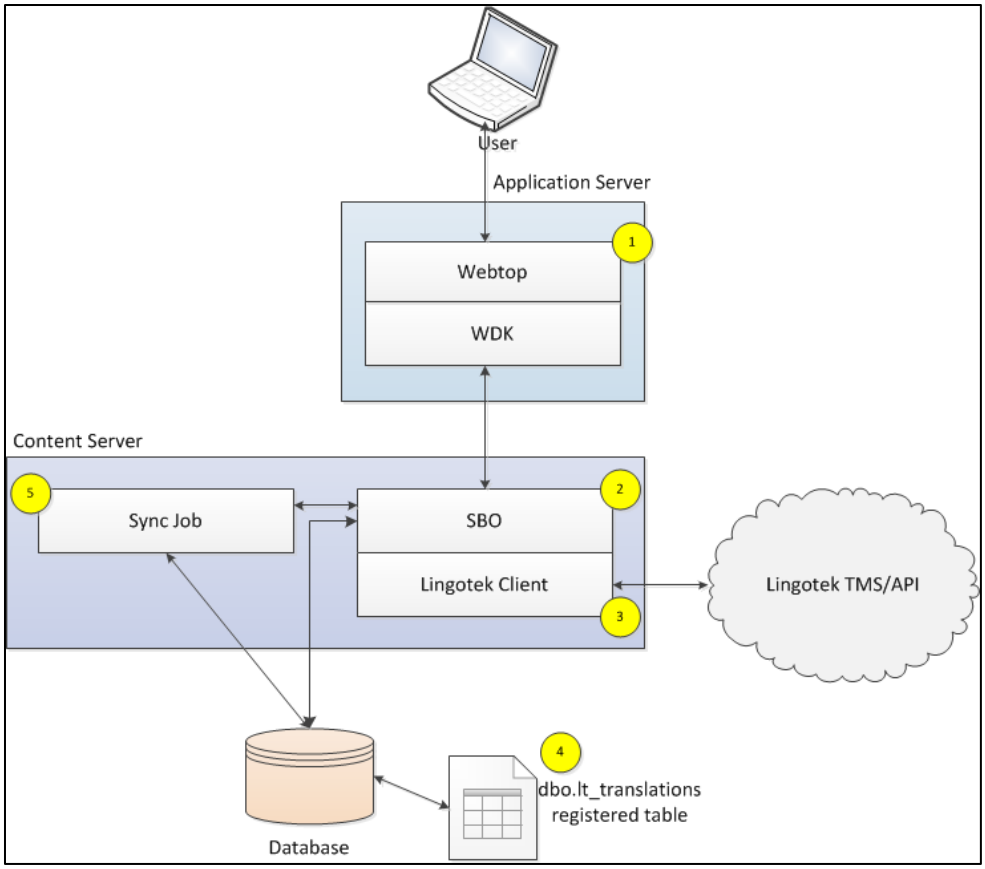


Figure 1: Software Architecture

The primary components of the solution are:

1. Webtop Customization – A Webtop customization provides a simple user interface (UI) for the user to select the target language, etc. for the translation. The Webtop customization was created using Documentum's web API (the Web Development Kit, WDK) and is discussed in more detail in Section 3.4.
2. Service-based Object – An SBO provides all the necessary translation services in Documentum, and is available from the WDK, the Documentum API (DFC), other BOF objects, workflows, and lifecycles. The SBO is discussed in detail in Section 3.2.
3. Lingotek Client – The Lingotek client is a Java class that implements the Lingotek RESTful API and provides the necessary interfaces and communication between the SBO and the Lingotek TMS. The Lingotek client and API are discussed in Section 3.1.
4. Registered Table – The registered table is a key interface between the original document and its translation(s). It also acts as an audit trail for the translation process by

recording the status of each document sent for translation. Details of the registered table are discussed in Section 3.3.

5. Synch Job – The synch job is the mechanism that completes the asynchronous translation process. The synch job uses the information recorded in the registered table and the logic in the SBO to retrieve document translations from the Lingotek TMS and import them into Documentum. Details of the synch job are discussed in Section 3.5.

2.2.2 Translation Process

Document translations occur in two separate processes: the Translation Request and the Translation Retrieval. These two processes are discussed in the following sections.

2.2.2.1 Translation Request

The translation process begins when the user requests a translation for a document. Section 3.4 discusses the Webtop UI components to enable this. When a translation request is made, Webtop instantiates the `LtTranslateSBO` SBO, which in turn, authenticates to the Lingotek TMS. Webtop then requests a translation from the SBO, which exports the content file from the Documentum repository, uploads it to the Lingotek TMS, and starts the translation workflow in the TMS. A successful upload to Lingotek returns a document Id to the SBO. The SBO updates the registered table with the document's metadata and the Lingotek document Id, and then returns the document Id to Webtop for display to the user. This logic is depicted in the sequence diagram in Figure 2.

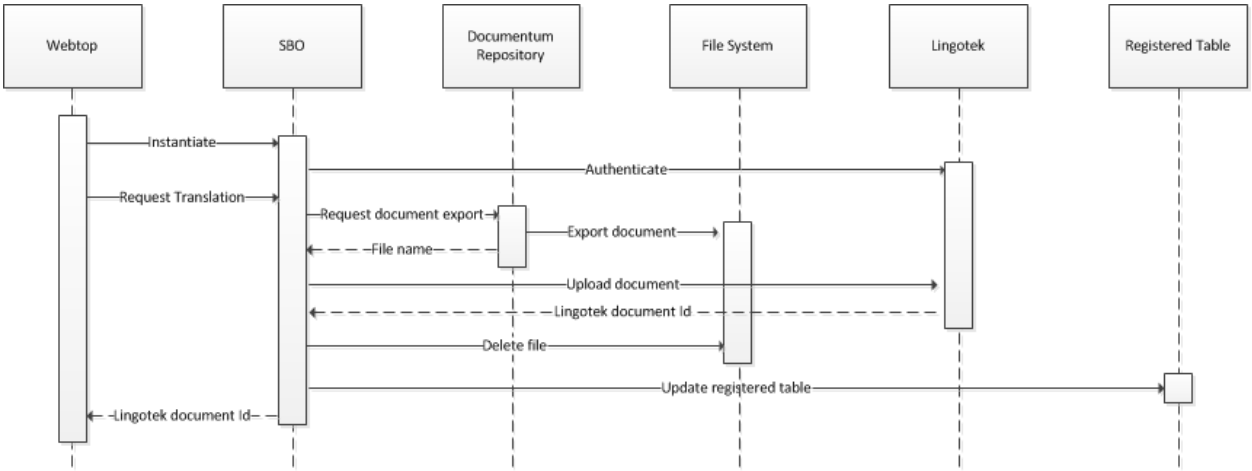


Figure 2: Request Translation Sequence

2.2.2.2 Translation Retrieval

The second step in the translation process is retrieval of the translation from the Lingotek TMS, once the translation is completed. This step of the process occurs asynchronously to the request for translation and is performed by the `LtTranslateSyncJob` job (see Section 3.5). When the job launches, it queries the `lt_translations` registered table (see Section 3.3) for any newly entered or pending translation requests. The job then instantiates the `LtTranslateSBO` SBO and tries to retrieve the translations from Lingotek. If a translation is successfully retrieved from the Lingotek TMS, it is imported into Documentum and related to the original document using the built-in `DM_TRANSLATION_OF` relation. The job updates the registered table to include the `r_object_id` of the newly imported translation. The final step in the process is to notify the user via an Inbox message that the translation is complete. This logic is depicted in the sequence diagram in Figure 3.

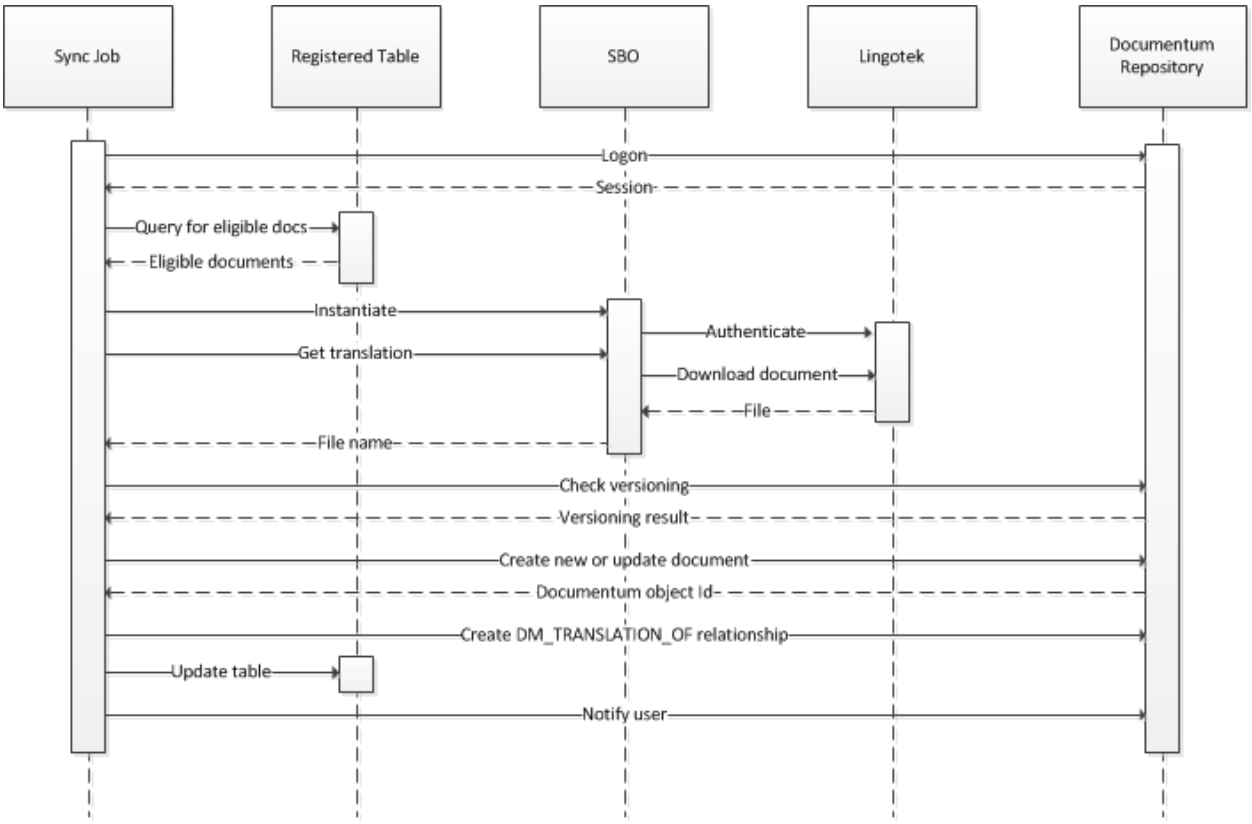


Figure 3: Retrieve Translation Sequence

3 Implementation

As shown in Figure 2 and Figure 3, the SBO plays a key role in both the request and retrieval of translations. Recall that this was the goal: to consolidate all interaction with the Lingotek TMS in the SBO, thus making it available to all parts of the Documentum architecture. This section discusses the implementation of the SBO and other key components of the integration between Documentum and Lingotek.

3.1 Lingotek Client and API

The Lingotek API is presented as a set of RESTful web services[8]. These web services are encapsulated in the `LingotekClient` class[9] to ease integration with Documentum. The Lingotek API utilizes OAuth[10] security framework for authentication of clients, and JSON[11] as its data-interchange format. The `LingotekClient` class simply encapsulates these technologies, consolidates their raw functionality into logical components, and presents them for use to the SBO. Details of this class are not included here[12].

3.2 LtTranslateSBO SBO

The `LtTranslateSBO` SBO provides all of the language translation services required by Webtop, the synch job, and all other components of the Documentum infrastructure. The SBO is implemented as an Interface and an Implementation class as depicted in Figure 4.

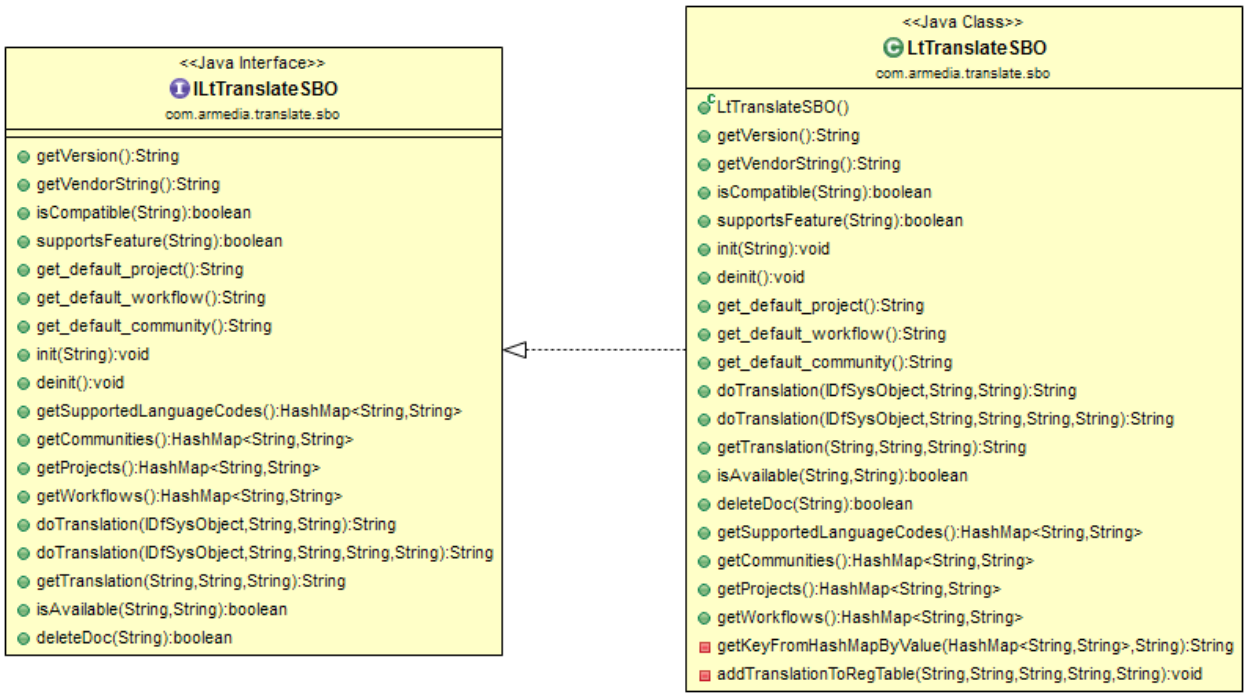


Figure 4: LtTranslateSBO Class and Interface

The SBO's public methods are:

- *get_default_project()* – returns the default Lingotek TMS project loaded by the SBO initialization process (see Table 1 in Section 3.2.1).
- *get_default_workflow()* – returns the default Lingotek TMS workflow loaded by the SBO initialization process (see Table 1 in Section 3.2.1).
- *get_default_community()* – returns the default Lingotek TMS community loaded by the SBO initialization process (see Table 1 in Section 3.2.1).
- *getSupportedLanguageCodes()* – returns a list of all languages (as IETF language codes)[13] supported by the Lingotek TMS.
- *getCommunities()* – returns a list of all Lingotek TMS communities accessible to the user.
- *getProjects()* – returns a list of all Lingotek TMS projects accessible to the user.
- *getWorkflows()* – returns a list of all Lingotek TMS workflows accessible to the user.
- *doTranslation(IDfSysObject document, String source, String target)* – translates the content of the `document` from the `source` language to the `target` language, and returns the Lingotek document Id. This method uses the default project and workflow to perform the translation.
- *doTranslation(IDfSysObject document, String project, String workflow, String source, String target)* – translates the content of the `document` from the `source` language to the `target` language, and returns the Lingotek document Id. This method uses the `project` and `workflow` passed as arguments to accomplish the translation.
- *getTranslation(String documentId, String locale, String path)* – retrieves the translation from the Lingotek TMS specified by `documentId` and `locale`, and saves it to the path contained in the `path` argument.
- *isAvailable(String documentId, String locale)* – returns `true` if the translation indicated by the `documentId` and `locale` is available to be retrieved from the Lingotek TMS (i.e., the translation is completed); else `false`.
- *deleteDoc(String documentId)* – deletes the document specified by the `documentId` from the Lingotek TMS, and returns `true`; else `false` if the document cannot be deleted.

The following sections provide more detail regarding initialization of the SBO, and the contents and use of the `lt_trans_config` configuration object.

3.2.1 SBO Initialization

After instantiating the SBO, the *`init()`* method must be called to initialize the translation services. The *`init()`* method queries the Documentum repository for a configuration object (see Section 3.2.2), loads default values, authenticates the user with the Lingotek TMS, and loads initial values from the Lingotek TMS. Figure 5 depicts the SBO's initialization process.

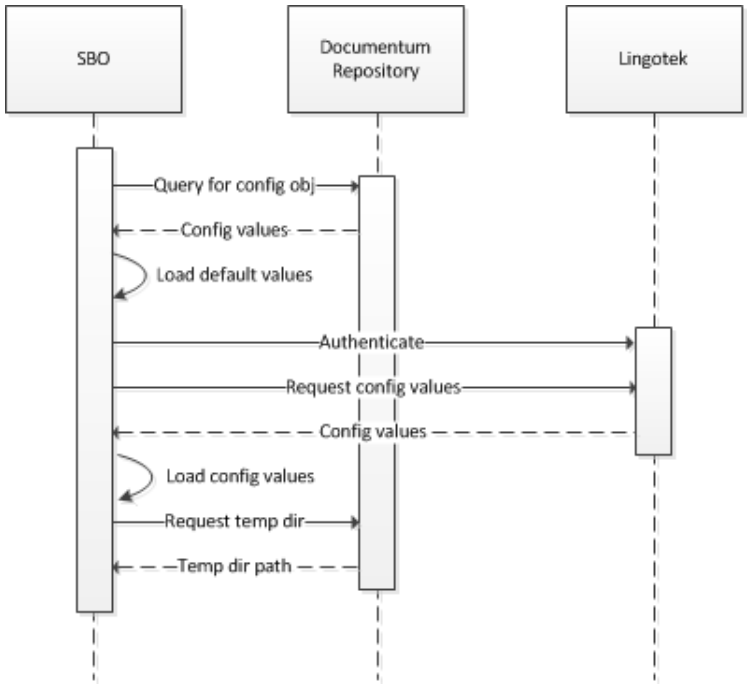


Figure 5: SBO Initialization Sequence

The SBO's initial values are loaded from two different places: the `lt_trans_config` object in the Documentum Content Server, and the Lingotek TMS user's profile. The initial values are described in Table 1.

Value	Source	Purpose
Bearer Token	`lt_trans_config` object	The bearer token is the OAuth security token issued by Lingotek when you create an account[14]. This token grants you access to your communities, projects, workflows, and documents in the Lingotek TMS[15].
Host	`lt_trans_config` object	This is the URL to the Lingotek RESTful web service API.
Default Community	`lt_trans_config` object	(Optional) If present, the SBO will use this community definition by default.
Default Project	`lt_trans_config` object	(Optional) If present, the SBO will use this project definition by default.
Default Workflow	`lt_trans_config` object	(Optional) If present, the SBO will use this workflow definition by default.
Locales	Lingotek TMS	A list of all supported languages as IETF language codes.
Communities	Lingotek TMS	A list of all communities accessible to user.
Projects	Lingotek TMS	A list of all defined projects accessible to user.
Workflows	Lingotek TMS	A list of all defined workflows accessible to user.

Table 1: LtTranslateSBO Configuration Values

The values retrieved from the `lt_trans_config` object are the authentication and default values used by the SBO. The bearer token and host values are necessary to log in to the Lingotek TMS. The default community, project, and workflow values are used to pre-populate UI controls (see Section 3.4). The projects, communities, and workflows are lists of available values obtained from the Lingotek TMS. These lists are also used to populate UI controls.

3.2.2 Configuration Object

The `lt_trans_config` object is a singleton object in the repository and holds the initial configuration information used by the `LtTranslateSBO` SBO. The `lt_trans_config` object is a subtype of `dm_sysobject`. Table 2 contains the attribute definitions for the object type.

Parent Type: dm_sysobject		
Attribute	**Data Type**	**Purpose**
`bearer_token`	String(40)	The Lingotek OAuth security token used by the SBO to authenticate with the Lingotek TMS.
`host`	String(64)	The URL for the Lingotek RESTful web service API; usually `http://cms.lingotek.com/api`.
`default_community`	String(40)	The name of the default community to be used by the SBO.
`default_project`	String(40)	The name of the default project to be used by the SBO.
`default_workflow`	String(40)	The name of the default workflow to be used by the SBO.
`object_name`	String(255)	(Inherited from `dm_sysobject`). As a best practice, name this object using the name of the repository followed by "-lt_config". For example: `repo1-lt_config`.

Table 2: lt_trans_config Object Attributes

3.3 Registered Table

The registered table, `lt_translations`, is the integration point between the actions started in Webtop by the user, and the completion of the translation process by the synch job. After a document has been successfully transferred to the Lingotek TMS for translation, the SBO inserts metadata for the document into this table. The `LtTranslateSyncJob` job then reads

the rows of the table to determine which translations to retrieve from the Lingotek TMS (see Section 3.5). Once translations have been successfully retrieved from the Lingotek TMS and imported into Documentum, the job updates the rows of this table to reflect each document's status.

The `lt_translations` table schema is described in Table 3.

Column Name	Data Type	Purpose
`source_obj_id`	`char(16)`	The `r_object_id` of the source document for translation. This is the document sent to Lingotek.
`source_locale`	`char(8)`	The IETF language code representing the language of the source document.
`target_locale`	`char(8)`	The IEFT language code representing the language to translate the source document into.
`translation_obj_id`	`char(16)`	The `r_object_id` of the translated document after it has been retrieved from the Lingotek TMS and imported into Documentum.
`lt_doc_id`	`char(40)`	The Lingotek document Id for the document after it has been uploaded to the Lingotek TMS.
`translation_status`	`char(12)`	The status of the translation. Possible values are: `SUBMITTED, PENDING, ERROR, COMPELTE`.
`update_date`	`datetime`	The date and time the row was last updated.
`remark`	`char(64)`	A remark about the processing of this document or translation. This column can contain versioning information or additional error information.

Table 3: lt_translations Table Scheme

Figure 6 depicts a portion of the `lt_translations` table. Notice the `SUBMITTED` row versus the `COMPLETE` rows. In the normal course of operating, the SBO will insert values for `source_obj_id`, `source_locale`, `target_locale`, `lt_doc_id`, `translation_status`, and `update_date` after sending a document to the Lingotek TMS for translation. This state is exemplified by the row in Figure 6 with a `SUBMITTED` status. When the sync job completes the translation process by retrieving a translation from the Lingotek TMS and importing it into the Documentum repository, the `translation_obj_id`, `translation_status` and `update_date` columns are updated as exemplified by the rows in Figure 6 with a `COMPLETE` status.

source_obj_id	source_locale	target_locale	translation_obj_id	lt_doc_id	translation_status	update_date
0901e4538000e9aa	en-US	es-ES	0901e4538001fe96	8733146a-ebb3-4605-a00c-b78bf9ff7d1c	COMPLETE	2014-08-14 15:06:02.000
0901e4538000e9aa	en-US	fr-FR	0901e4538002253b	efcbc629-6f4e-4bbb-93c9-18280b64eda2	COMPLETE	2014-08-21 15:28:16.000
0901e4538000e9aa	en-US	aa-DJ		9228ffce-829f-419a-8fae-0d305722623e	PENDING	2014-08-21 15:28:18.000
0901e4538000e9aa	en-US	es-ES		4843937f-d0bc-456a-8a14-b48c8fe87e52	SUBMITTED	2014-08-21 15:28:19.000

Figure 6: lt_translations Example

The `PENDING` row indicates a translation was not ready for retrieval from the Lingotek TMS at the time the job ran. It will be retrieved at the next scheduled run of the job. The `remark` column (not depicted in Figure 6 for brevity) may be updated with additional information regarding the translation submission, status, or retrieval at any time.

3.4 Webtop Customization

In the scenario described in Section 2.1, the user initiates a translation from Webtop. To achieve this, several simple Webtop components were created. The following sections describe these components.

3.4.1 Menu Customization

A Translation menu was added to the Webtop Tools menu (see Figure 7), which includes two sub-menus: Request Translation… and Show Dashboard. The Request Translation… menu option opens the Request Translation dialog box discussed in Section 3.4.2, and the Show Dashboard menu option opens the Dashboard dialog discussed in Section 3.4.3. These menu options were implemented using the usual `*_menu_config.xml`, `*_component.xml`, and `*_actions.xml` files in the Webtop `/custom` folder hierarchy[16].

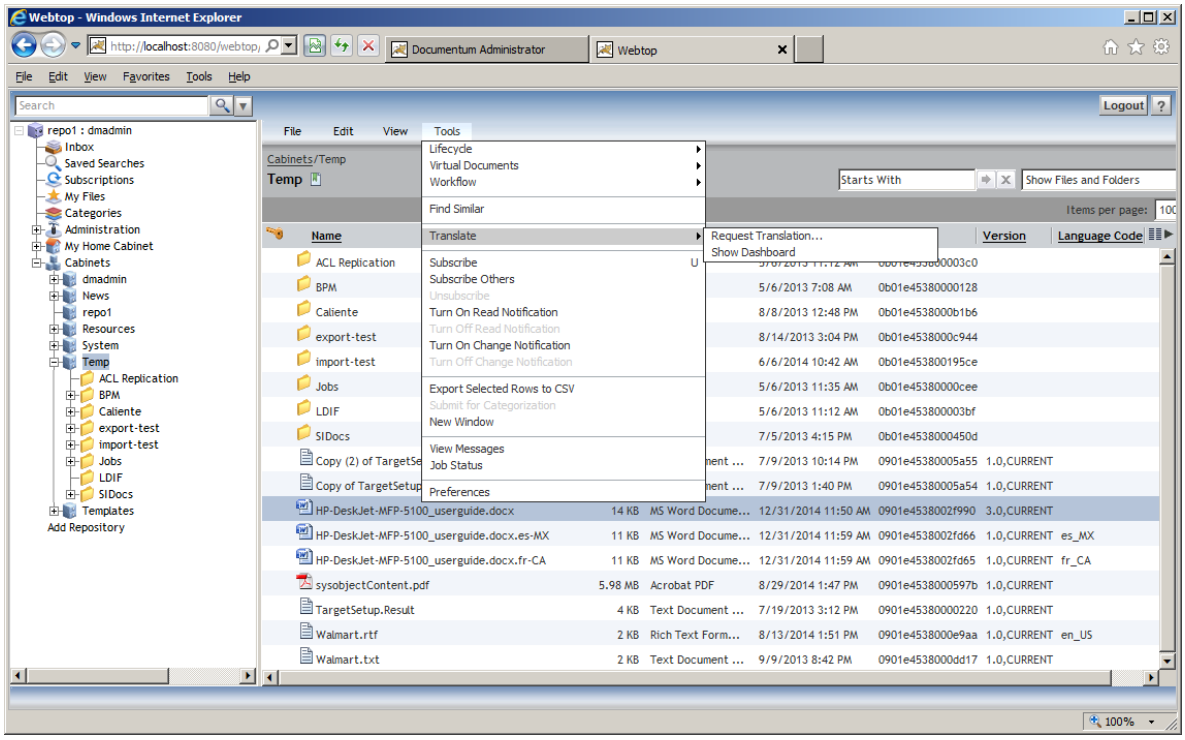


Figure 7: Translation Menu

3.4.2 Request Translation Dialog

The Request Translation dialog, depicted in Figure 8, is the integration's primary Webtop UI and utilizes the SBO's methods discussed in Section 3.2.

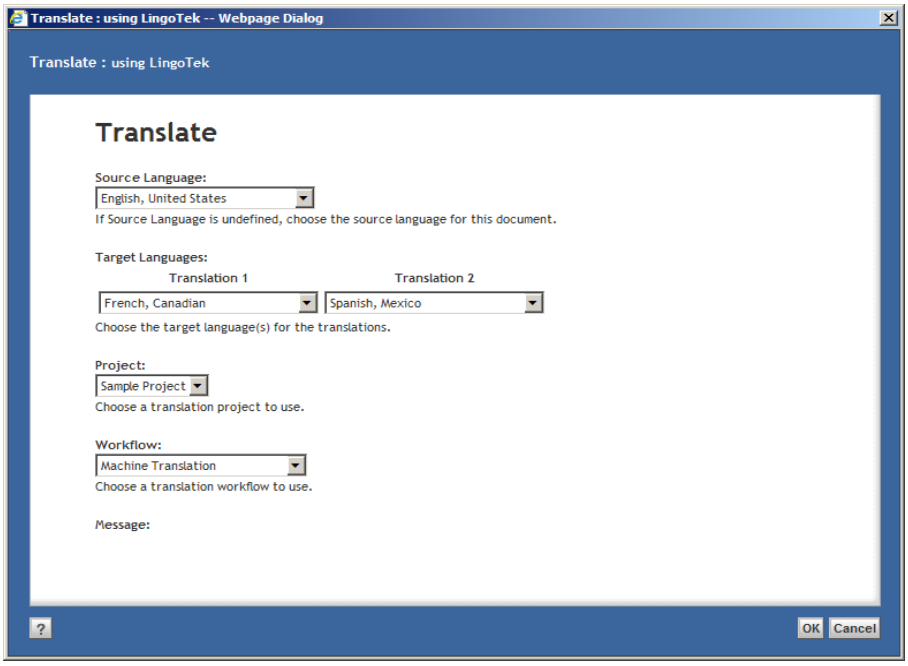


Figure 8: Request Translation Dialog

When the user selects a document to translate in Webtop and then selects the Tools → Translation → Request Translation… menu option, Webtop passes the selected object's Id to the Request Translation dialog. The class file backing the dialog checks to ensure the object contains content and then begins to initialize the dialog's controls for the user. This sequence of events is depicted in Figure 9.

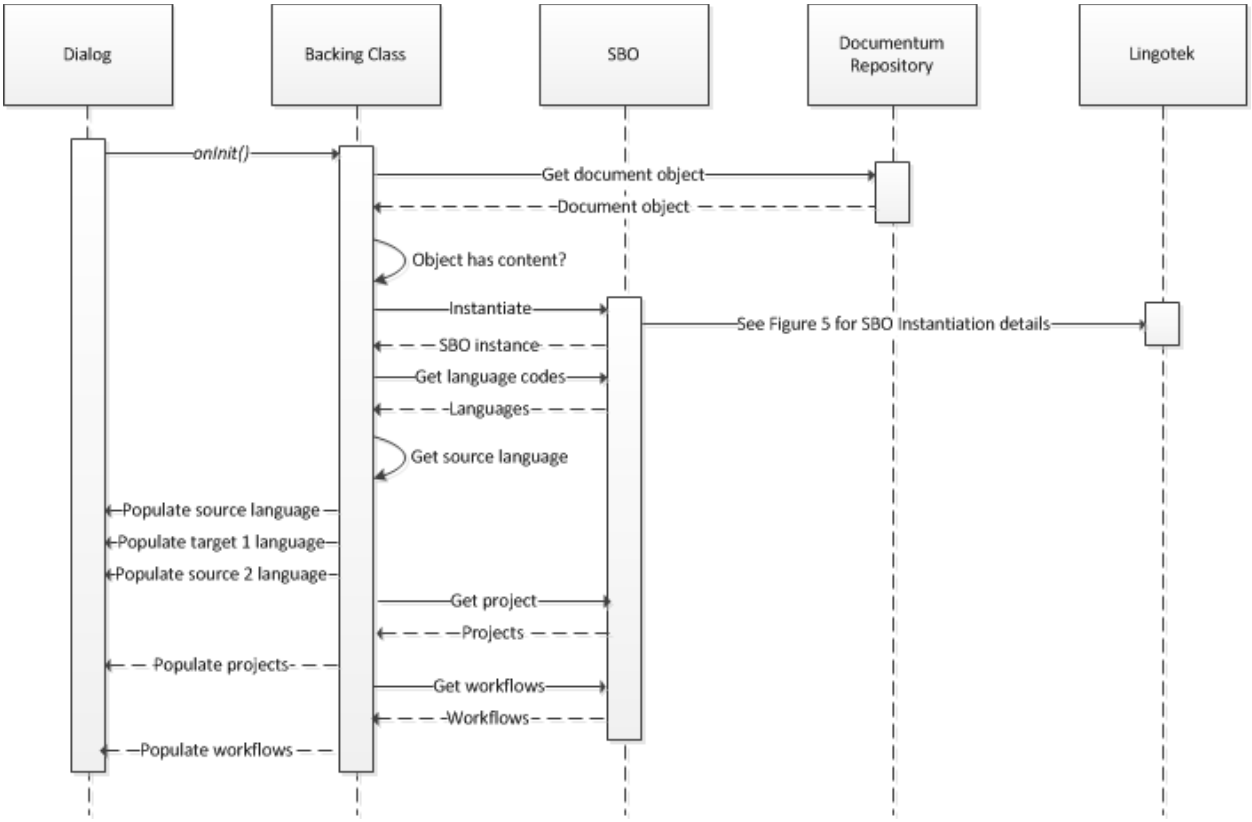


Figure 9: Request Translation Dialog Initialization Sequence

First, the backing class instantiates and initializes the SBO (as described in Section 3.2.1), and then proceeds to load all of the dropdown controls with values retrieved from the Lingotek TMS. The Source Language, Translation 1, and Translation 2 dropdowns are loaded with all of the languages supported by Lingotek using the SBO's *getSupportedLanguageCodes()* method. If the object selected by the user has a source language defined in its dm_document.language_code attribute, the Source Language dropdown control is disabled and the language code defined on the object is used as the source language. The Project and Workflow dropdown controls are similarly filled using the SBO's *getProjects()* and *getWorkflows()* methods respectively. If default values are defined in the lt_trans_config object (see Section 3.2.2), these controls are defaulted to those values

(using the SBO's *get_default_project()* and *get_default_workflow()* methods). Note that once the SBO is instantiated and initialized, the dialog screen can be initialized without having to access the Lingotek TMS again.

Once the user has made their selections for target language(s), etc. and clicks the OK button, the dialog's class file validates the selections and requests the translation using the SBO's *doTranslation()* method. The SBO returns the translation's Lingotek TMS document Id to the dialog which displays it using Webtop's `MessageService`. The SBO is de-initialized and the Request Translation dialog window is closed. The sequence of events for submitting documents to Lingotek for translation was previously detailed in the sequence diagram in Figure 2, Section 2.2.2.1.

3.4.3 Dashboard Dialog

The Dashboard dialog, depicted in Figure 10, provides a snapshot (the top 200 rows) of the `lt_translations` table, discussed in Section 3.3, in addition to a few summary statistics. It is provided as a simple way to monitor the operation of the integration and check the status of translation requests. This dialog was built using the WDK, Java, and the DFC; it does not utilize the SBO.

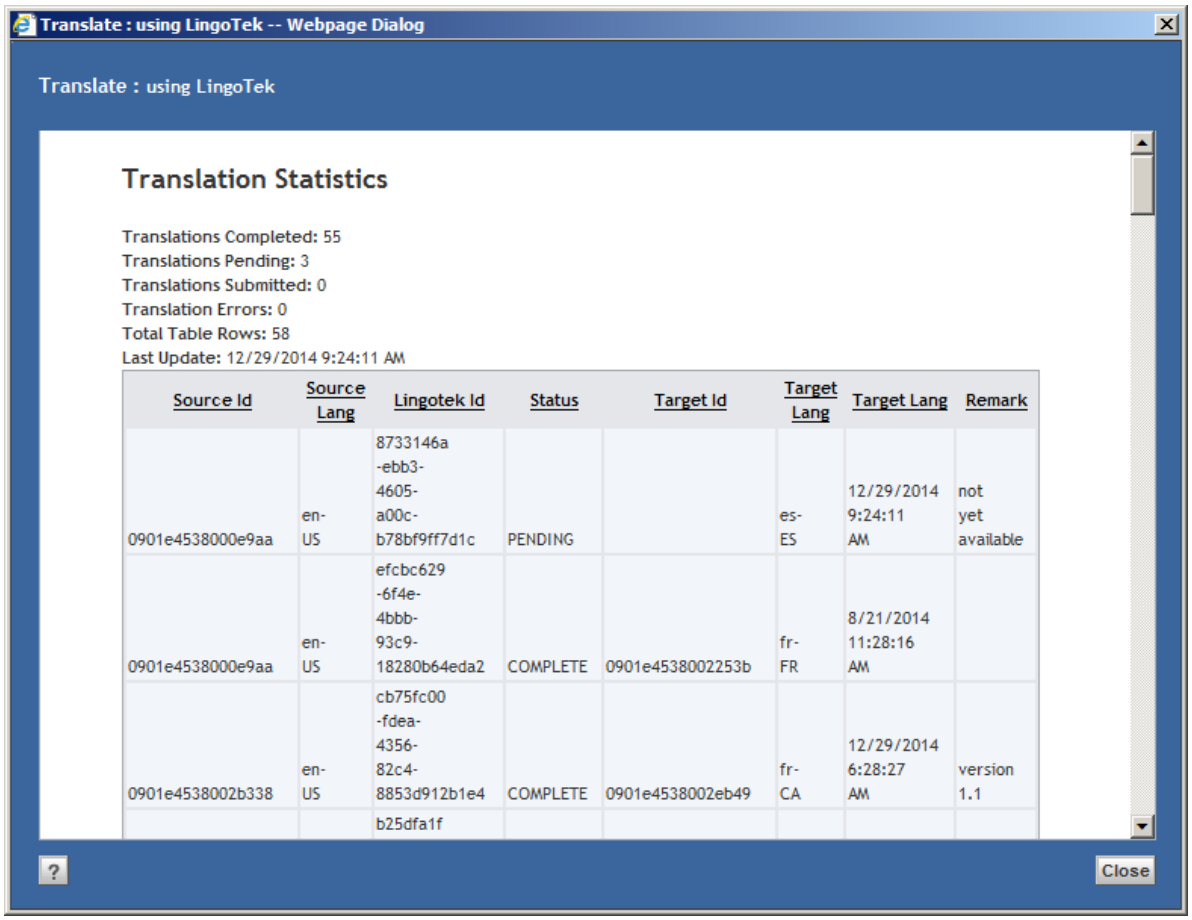


Figure 10: Dashboard Dialog

3.5 LtTranslateSyncJob Job

The first half of the translation process was to submit documents from Documentum to Lingotek's TMS for translation; these steps were discussed in Section 3.4. The second half of the translation process is to retrieve the translations from Lingotek when they are available and import them into Documentum. Because the translation process is asynchronous, it is not practical for the SBO to wait for each translation to complete when it is submitted. Therefore, a Documentum job completes the work. The `LtTranslateSyncJob` job runs frequently[17], polling the `lt_translations` registered table and the Lingotek TMS for completed translations to be retrieved and imported into Documentum.

When the `LtTranslateSyncJob` job starts, it queries the `lt_tranlations` table for all Lingotek document Ids that have a status of `SUBMITTED` or `PENDING` and then tries to retrieve them from the Lingotek TMS using the SBO methods discussed in Section 3.2. Once retrieved, each document is imported into Documentum one of two ways: as a new document or as a

version of an existing document. The heuristic to make this determination is discussed in Section 3.5.2.

Once imported, the translation's `r_object_id` is written into the `lt_tranlations` table, its status is updated to `COMPLETE`, and the `update_date` is updated. The translation is then related to the original document using a built-in, Documentum relationship, `DM_TRANSLATION_OF` (see Section 3.5.3). This enables simple correlation between original documents and their various translations. The final step in the process is to alert the user that the translation is available. This step is discussed in Section 3.5.4.

The sequence of events for retrieving documents from the Lingotek TMS was previously detailed in the sequence diagram in Figure 3, Section 2.2.2.2.

3.5.1 Translation Naming Convention

As each translation is retrieved by the synch job from the Lingotek TMS, it is saved to the same location in the Documentum repository as the original source file, and the translation's locale code (i.e., IETF language code) is appended to the object's name for easy identification. See the example in Table 4.

Source Document	Translation	Language
HP-DeskJet-MFP-5100_userguide.docx	HP-DeskJet-MFP-5100_userguide.docx.es-MX	Spanish-Mexico
HP-DeskJet-MFP-5100_userguide.docx	HP-DeskJet-MFP-5100_userguide.docx.fr-CA	French-Canada

Table 4: Naming Convention Example

3.5.2 Versioning Logic

The logic for determining how a translation is imported into Documentum briefly works like this. The Documentum repository is queried to see if a translation having the same source document and language code already exists. If not, the translation is imported as a new document.

If an existing translation is found, the version tree of the source document is evaluated and paired with accompanying versions of the translation until the latest version of the translation is found. The latest version of the translation is then checked out, the imported document is

attached as the new content file, and the translation checked in. Versioning of the translation being checked in is determined as follows: If the difference between the source document's current version and its last version is a major version, the translation is checked in as a major version. If the difference is a minor version, the translation is checked in as a minor version. If the current version of source document already has a translation (i.e., the source document was checked in as the same version), the translation is checked in as the same version of the corresponding translation. Note that the source document and its translation will *not* always have the same version numbers (e.g., 1.0 -> 1.0, 1.1 -> 1.1, etc.); however, the translations will maintain version parity with each version of the source document used to create it. See Table 5 for an example.

Source Version		Translation Version
v1.0	→	v1.0
v1.1		
v1.2		
v1.3	→	v1.1
v2.0	→	v2.0
v2.0	→	v2.0

Table 5: Versioning Logic Example

In Table 5, version 1.0 of the source document was used to create translation version 1.0. The source document was then edited twice (v1.1 and v1.2), but no corresponding translations were made for these documents. Version 1.3 was checked in and a translation requested. According to the heuristic, the source document version tree was walked looking for the latest corresponding translation version. Translation v1.0 was found, so this translation was checked in as v1.1 of the translation. Version 2.0 or the source document produced version 2.0 of the translation because the difference between the last source version (v1.3) and the current version was a major version. Finally, in the last row, table version 2.0 of the source document

was checked in as the same version which resulted in version 2.0 of the translation being checked in as the same version.

As you can see, the version numbers of the corresponding documents and translations do not always match (source v1.3 → translation v1.1); however, the version tree of each document and translation is intact. Additionally, version pairing information is included in the description field of each `DM_TRANSLATION_OF` relationship when it is formed to help identify which translation version corresponds to which source version, for example:

```
parent en-US v1.3 -> child fr-CA v1.1
```

3.5.3 DM_TRANSLATION_OF Relationship

Documentum provides a built-in relationship named `DM_TRANSLATION_OF` specifically for denoting that one document is a translation of another. This provides a very convenient way for the Documentum-Lingotek integration to manage and relate source documents and translations without involving additional code or customizations. In addition, because the `DM_TRANSLATION_OF` relationship is a known Documentum relationship, it is easy to view all translations of a particular source document directly from Webtop. Figure 11 depicts the Webtop View Relationships screen. Though this is not the easiest screen to interpret, it does contain the necessary information to indicate that the selected document (HP-DeskJet-MFP-5100_userguide.docx) is the parent of two other documents involved in a `DM_TRANSLATION_OF` relationship. Those two documents are the Spanish and French translations, as indicated by their naming convention.

Relating back to a scenario described in Section 2.1, another Documentum job or workflow could leverage these relationships to publish new content to a website when it becomes available.

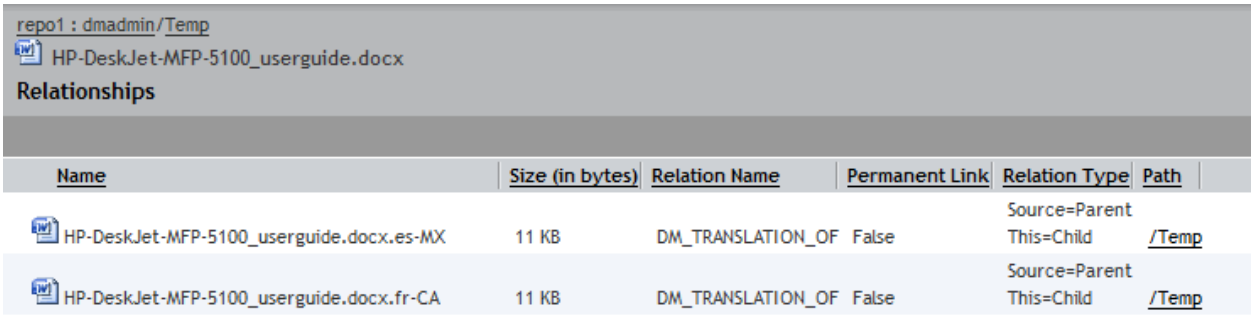


Figure 11: Webtop View Relationships Screen

As an aside, it would be nice if the Webtop View Relationships screen display the version label and the relationship description for each document as well.

3.5.4 User Notification

The final step in the process of retrieving translations from the Lingotek TMS and importing them into the Documentum repository is to alert the original document's owner that the translation is available. This is accomplished by sending an Inbox notification to the document's owner. See Figure 12 for an example of an Inbox notification.

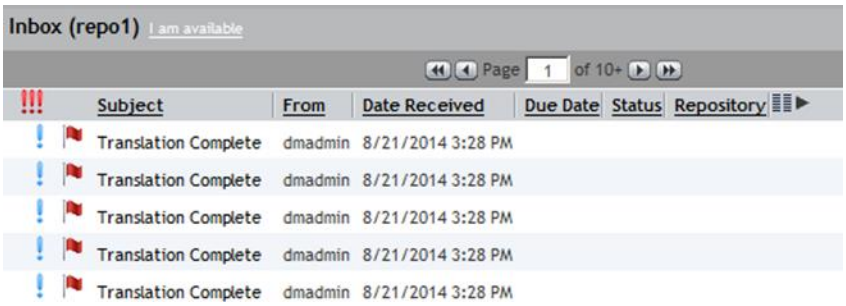


Figure 12: User Notification via Inbox

3.6 Log Files

In addition to using the Dashboard dialog discussed in Section 3.4.3, privileged users (i.e., system administrators) can also track translations using three log files: the Webtop application server log file, the Documentum Java Method Server (JMS) log file, and the job report attached to the `LtTranslateSyncJob` job object. Each of these log files are discussed in the following sections.

3.6.1 Application Server Log File

The application server log file records the interactions among Webtop, the SBO, and the Lingotek TMS. See Figure 13 for an example.

```
***** start LtTranslate dialog java class *****
objectId=0901e4538002b338
***** init LtTranslateSBO (c) 2014 Armedia, LLC v0.a *****
bearer token=e52e740c-60b4-30ed-8876559cdb
host=http://cms.lingotek.com/api
src lang=en-US
project=Sample Project
workflow=Machine Translation
target lang1=fr-CA
Submitting document for translation...
downloading C:\Program Files\Apache Software Foundation\Tomcat
7.0\temp/0901e4538002b338-en-US.docx
deleted C:\Program Files\Apache Software Foundation\Tomcat
7.0\temp/0901e4538002b338-en-US.docx
Translation submitted successfully, doc Id: cb75fc00-fdea-4356-8853d912b1
***** deinit LtTranslateSBO *****
***** end LtTranslate dialog java class *****
```

Figure 13: Application Server Log File Snippet

This log file provides a good summary of the translation specifications set on the Request Translation dialog, as well as the key interactions between the SBO and the Lingotek TMS.

3.6.2 Java Method Server Log File

Similar to the information captured in the application log file discussed in Section 3.6.1, information captured in the JMS log file is equally helpful in observing the interaction among the `LtTranslateSyncJob` job, the SBO, and the Lingotek TMS. See Figure 14 for an example.

```
INFO [STDOUT] ***** Start LtTranslateSyncJob ^*****
INFO [STDOUT] Using C:\Windows\TEMP\ for temporary storage.
INFO [STDOUT] ***** init LtTranslateSBO (c) 2014 Armedia, LLC v0.a *****
INFO [STDOUT] bearer token=e52e740c-60b4-30ed-8876559cdb
INFO [STDOUT] host=http://cms.lingotek.com/api
INFO [STDOUT] processing 0901e4538002b338 (en-US => fr-CA) ff04098a-bb0e-
4d67-4f4247c93c, status:SUBMITTED
INFO [STDOUT] processing file C:\Windows\TEMP\0901e4538002b338-en-
US.fr_CA.docx (LT Id=ff04098a-bb0e-4d67-4f4247c93c)
INFO [STDOUT] INFO: checked in 0901e4538002f2dc as version 1.1 of
0901e4538002b338
INFO [STDOUT] ***** deinit LtTranslateSBO *****
INFO [STDOUT] ***** end LtTranslate dialog java class *****
```

Figure 14: JMS Log File Snippet

As you can see from this log file snippet, a document that was submitted for translation was successfully retrieved from the Lingotek TMS and imported into Documentum as a version of a previously existing translation.

3.6.3 Job Report

The job report is attached to the Documentum job object after each run. The job report can be accessed from the Documentum Administrator (DA) client, by navigating to the Jobs node, right-clicking on the LtTranslationSyncJob job, and choosing View Job Report. Figure 15 contains a sample job report.

```
LtTranslateSyncJob log

Using C:\Windows\TEMP\ for temporary storage.
1. processing 0901e4538002f990 (en-US => fr-CA) 87f16e7e-d015-4087-
59af101169, status:SUBMITTED
   COMPLETE: 0901e4538002f990
2. processing 0901e4538002f990 (en-US => es-MX) e4affce4-450a-442c-
3d6bc2a217, status:SUBMITTED
   COMPLETE: 0901e4538002f990
Processed 2 translation requests

Job completed Successfully. Total run time was 5 sec.
```

Figure 15: Job Report

Historic job reports files can be found in the Documentum repository in the `/System/SysAdmin/Reports` folder. The `LtTranslateSyncJob` reports are versioned to reduce clutter in this directory.

4 Sample Uses of SBO

This section contains short snippets of code to demonstrate how the SBO, its methods, and its configuration values can be used.

4.1 SBO Instantiation

Instantiate the SBO using the standard SBO instantiation methodology.

```
IDfSessionManager sMgr = getSession().getSessionManager();
IDfClient client = getSession().getClient();
ILtTranslateSBO SBO = (ILtTranslateSBO)
   client.newService(ILtTranslateSBO.class.getName(),sMgr);
if(SBO == null)
   throw new Exception("Could not instantiate LtTranslate SBO.");
```

4.2 SBO Initialization

Before the SBO can be used to request or retrieve translations, it must be initialized.

```
SBO.init(getSession().getDocbaseName());
```

4.3 Request a Translation

To submit a document for translation, use code like this:

```
String LtDocId = SBO.doTranslation(sObj, "Sample Project",
   "Machine Translation", "en-US", "fr-CA");

   if ((LtDocId != null) && (! LtDocId.trim().isEmpty()) ) {
        System.out.println("Translation submitted successfully, doc
             Id: " + LtDocId);
```

Or, using the SBO's default configuration values:

```
String LtDocId = SBO.doTranslation(sObj, "en-US", "fr-CA");

   if ((LtDocId != null) && (! LtDocId.trim().isEmpty()) )
        System.out.println("Translation submitted successfully, doc
             Id: " + LtDocId);
```

The result of a successful translation request would be:

```
Translation submitted successfully, doc Id: cb75fc00-fdea-4356-
8853d912b1
```

4.4 Retrieve a Translation

To retrieve a translation from the Lingotek server, use code like this:

```
if (SBO.isAvailable("cb75fc00-fdea-4356-8853d912b1", "fr-CA")) {
   String fullFilename = SBO.getTranslation("cb75fc00-fdea-4356-
8853d912b1", "fr-CA", "C:\temp");
   System.out.println("Translation file: " + fullFilename);
}
```

The result of this successful retrieval of a translation would be:

```
Translation file: C:\Windows\TEMP\0901e4538002b338-en-US.fr_CA.docx
```

4.5 WDK Example

This example demonstrates how to load a WDK dropdown list with all available Lingotek workflows.

```
ArrayList<String> vals = new ArrayList<String>(
   SBO.getWorkflows().values());

for (String v : vals) {
   Option opt = new Option();
   opt.setLabel(v);
   opt.setValue(v);
   WorkflowDDL.addOption(opt);
}

// set default value
if (SBO.get_default_workflow()!= null &&
   ! SBO.get_default_workflow().trim().isEmpty())
   WorkflowDDL.setValue(SBO.get_default_workflow());
```

4.6 SBO Deinitialization

The SBO should be de-initialized after its use.

```
SBO.deinit();
```

5 Conclusion

As previously stated, the *process* of creating and managing translations of documents can be tedious. Services like those provided by Lingotek help improve this process by providing a platform to produce and collaborate on the production of translations. However, integration with a content management system like Documentum on the "source" side of the process completes the picture. With the integration discussed in the article, users are able to create and manage content in a Documentum repository using all of the power and features Documentum offers, and integrate their translation production process with the power and features of the Lingotek TMS. Together, these two systems complement each other to provide the end user a seamless translation production experience. In addition, this article discussed ideas for naming files to assist with identifying translations intuitively, a heuristic for managing and syncing versions of source files with translations, and how to leverage Documentum's built-in use of relationships to identify translations of specific documents for publication.

Additionally, I demonstrated that I was able to encapsulate all interaction with the Lingotek TMS in an SBO, thus making translation services available to Webtop, jobs, workflows, lifecycles, D2, and xCP.

6 Source Code

The Documentum-Lingotek integration discussed in this article is a proof-of-concept and is not production ready. There are portions of the code that have not been thoroughly tested (e.g., version management) and portions that could be refactored for better performance (e.g., submission of multiple translations of the same source document). If you are interested, I will provide the source code upon request; please send me an email.

7 Dedication

This article is dedicated to the memory of my long-time friend and colleague, Brian Yasaki (1956 – 2014). In addition to being a great friend, colleague, and officemate for almost 10 years, Brian was the beta tester for ideas such as this one. He thoroughly tested my ideas and code, and made them vastly better than I could by myself. Rest in peace, my friend, you are missed.

8 About the Author

Scott Roth is a Director of Technology at Armedia, LLC (http://www.armedia.com). Armedia is a systems integration firm that specializes in Enterprise Content Management solutions for the federal government and commercial customers. Armedia is CMMI-3 certified, and provides full software development lifecycle services, as well as virtualization and cloud services. Scott has been working with Documentum since 1997 and is the author of *A Beginner's Guide to Developing Documentum Desktop Applications* (ISBN: 0-595-33968-9). Scott is a certified EMC Proven Professional, a member of the EMC Elect 2014 and 2015 classes, and a previous EMC Proven Professional Knowledge Sharing published author (2014).

[1] John Deere riding lawn mower warranties: http://www.deere.com/en_US/products/equipment/riding_mowers/lawn_tractors/100_series/d105/d105.page. Click the "Support" tab.

[2] The IRS provides several forms in Spanish (look for the "SP" at the end of the form number) here: http://apps.irs.gov/app/picklist/list/formsPublications.html

[3] For example, HP user manuals for its DeskJet All-in-One MFP line: http://h10025.www1.hp.com/ewfrf/wc/manualCategory?cc=us&dlc=en&lang=en&lc=en&product=5330821&. In this case, as single manual contains instructions in multiple languages.

[4] For example, Bank of America website in two different languages: English: https://www.bankofamerica.com, and Spanish: https://www.bankofamerica.com/homepage/overview.go?request_locale=es_US.

[5] "Documents" here could well be a web page (sans its surrounding framework and controls), or a PDF or some other rich document format.

[6] Lingotek currently has integrations with Microsoft SharePoint, Drupal, Alfresco, WordPress, and others. Learn more at http://www.lingotek.com/learn-more-about-lingotek-translation-network.

[7] Service-based Objects (SBOs) are part of Documentum's Business Object Framework (BOF). The BOF is designed to be a business logic customization layer that sits between the DFC and the Content Server. BOF customizations are available and enforced by all Documentum clients. For additional information regarding SBOs and the BOF, see the *EMC Documentum Foundation Classes Development Guide*.

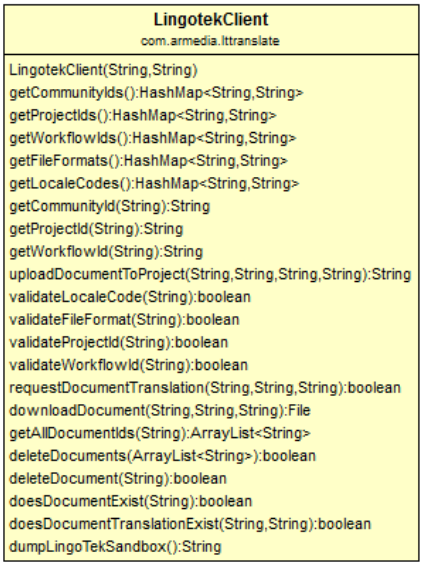
[8] See here for more details regarding the Lingotek RESTful API: http://devzone.lingotek.com/overview-5.

[9] The LingotekClient is based upon sample code provided here: http://devzone.lingotek.com/code-examples/java.

[10] For more information regarding OAuth, see: http://oauth.net/.

[11] For more information regarding JSON, see: http://json.org/.

[12] If you are really interested, here is the LingotekClient class diagram.



[13] Supported languages are identified using IETF language tags, for example: en-US = English spoken in the United States, fr-CA = French spoken in Canada, and es-MX = Spanish spoken in Mexico. For more information regarding IETF language tags, see here: http://en.wikipedia.org/wiki/IETF_language_tag.

[14] Create a Lingotek developer account: http://devzone.lingotek.com/overview-5. Click the Register link.

[15] See here for more details regarding communities, projects, and workflows: http://devzone.lingotek.com/glossary.

[16] A rather dated but still good discussion of basic Webtop development can be found here: https://developer-content.emc.com/developer/componentexchange.htm#0900c3558096d555.

[17] For this proof-of-concept, the job is scheduled to run every two minutes. In reality, this time may be too frequent or infrequent depending upon the business scenario, volume and size or documents to translate, and computing resources.