TM Migration Process: Trial by Fire

Table of Contents

Preface

I am especially proud that I was the owner and pioneer of the TM Migration Process. Before I worked on this process, the translation memory (TM) files from acquired companies could not be integrated into Business Objects' system because of technical difficulties. I worked closely with Language Specialists in some phases of the process, but for the vast majority of the time I worked completely independently to find each of the technical problems before seriously examining them to find an appropriate solution.

If I could sum up the migration process in one word, it would be "problematic." Since all my work essentially involved troubleshooting problems, which I also had to discover for myself, I find it fitting to discuss each of the major problems which I discovered, dissected, and solved in order to complete the migration successfully. If at any time there is doubt as to how I found the problem initially, the answer is painstaking sanity checking; if at any time there is doubt as to how I found the solution, the answer is painstaking trial.

Appendix A is the final TM Migration Process from the intranet Wikipedia; initially, the process was merely the headings for each step.

Introduction

Translation Team Overview

The translation process is a difficult but integral part of the software cycle; despite its importance and embodiment of the global outreach of the company's software solutions, the translation team must not only struggle with the deadlines set by software developers and testers, but also the team must overcome the technical intricacies required to translate millions of words most efficiently and economically. Computers are very good at performing simple, repetitive tasks such as retrieving translations from an electronic dictionary, but a well-known limitation is computers' inability to comprehend the contextual meaning of a sentence due to the complex nature of linguistics.

While fully automated machine-translated text may become feasible in the future, only two options currently exist: use humans to fully translate the text or use machines (computers) to partially translate the text and use humans to complete the translations. Due to the sheer volume of translation work, the latter option is exercised at Business Objects.

Machine Translation

A translation memory (TM) can be visualized as a container of word pairs, which include a source string and a target string. Usually, the source string is English and the target string is the desired language to which the source should be translated. There may be several different implementations of a translation memory, each with various capabilities and metadata, but their common denominator is the word pairing.

Machine translation (MT) is achieved when translation software reads a passage of text, accesses the translation memory to find the source strings which match the text, and substitutes the source strings for the target strings. The aforementioned string metadata might include part of speech, context, and origin which are used to generate a matching percentage for each string or set of strings found in the passage of text. Given the complexity of the rules in translation software, it is possible for new content to be translated purely by MT; previously translated material, for example, the company's different products that share the same disclaimer, should be fully translated by MT, thus eliminating translation of the same content and reducing costs.

Enhancing Machine Translation

Trados is established translation software that is used to perform MT; this software allows a user to select a TM – there are many TMs which vary in content – for which the content is suitable to the desired translation text. There are some downfalls, which include keeping track of the many TMs (stored on Perforce), requiring the user to be proficient with the Trados software, and requiring the user to perform translation on his local machine.

Idiom WorldServer (WS) is a workflow solution that also provides a web interface to manage TMs. WS is not standalone software like Trados, so WS does not allow a single user to translate material on his own machine; rather, it is a server-based software package that

enables centralized, large-scale translations to be performed easily by many different users. The scalable and centralized advantages of WS give compelling reasons to move from the Trados-based translation methodology to the server-based one.

One of the challenges with moving to a new system is ensuring that everything which worked before the move works after the move. In the case of translations, the TMs are the most valuable assets. The majority of the moving work lies in setting up the new system and migrating the TMs from Trados to WS. As is the case with much software, each program uses its own file format and compatibility between programs can be problematic.

TM Migration Process

The Beginning

The migration process was initially a mere outline of general tasks which needed to be completed. There was a minimal amount of documentation on the complete process, save for the names of some steps that needed to be executed. A localization engineer at the development center in Paris had written conversion scripts in Perl that converted TM export (.tmx) files from Trados into WS-compatible .tmx files; these scripts were based on the company's own TMs and had already been executed to successfully migrate some of the company's own TMs to the new WS system. However, these scripts and the TM migration process had not yet been executed on non-company TMs. The following passages detail the creation of a process to successfully migrate non-company (legacy) TMs.

Roadblocks in Creating a New Process

Exporting the Files

In the past year, Business Objects acquired companies whose TMs had not yet been moved to WS. Their TMs were created by a newer version (version 7) of Trados and could not even be opened using Business Objects' older version (version 5) of Trados. Furthermore, the conversion scripts were based on the older version of Trados, thus the TMs had to be downgraded. Each of the external (legacy) TMs did have plain-text export files which represented the TMs in a .txt file instead of the usual .tmx format. The only workaround to the Trados version problem was to create a new, empty TM in version 5 for each of the TMs in version 7, import the version 7 TM data into the version 5 TM, and export the downgraded TM into a usable .tmx file.

Importing the data proved to be tricky as the empty version 5 TM did not accept Unicode-encoded files. This issue was discovered unfortunately only in the later steps of the migration process when special characters, like accented ones, appeared corrupt. There were two encoding problems with the files. First, each of the .txt files needed to be encoded in a Windows-friendly, language-specific format. For example, Unicode supported both Japanese and German characters, but the files for those two languages had to be encoded to two different Windows code pages (the operating system's encoding). Second, each of the .txt files included a header section with font definitions, called the preamble, which mapped Trados-specific tags to Unicode font styles. As these Unicode font styles were not supported during the import step, strings which used those styles and contained special characters displayed character corruptions.

To solve the first encoding issue, research was needed to find the suitable code page for each language. Then, each of the Unicode .txt files was converted to the appropriate code page using a conversion utility bundled with the Java Development Kit (JDK). This solution produced a completely lossless conversion which was essential in maintaining the quality of translations. Solving the second issue required the aid of a Perl script that summarized the Unicode font styles and tallied the number of occurrences of each style. Then, any Unicode font styles which were used by the text were changed to non-Unicode styles. After fixing both issues, the .txt files could be imported into the blank version 5 TM without further issues.

The fixes were verified via sanity checking, which involved opening a sample of the imported strings and comparing them to their pre-import state to ensure the highest quality and string integrity.

Patching the Scripts

Since the Perl conversion scripts were based on internal Trados 5 TMs, the scripts returned some errors while dealing with the legacy TMs which had been converted in the previous step. To fix the errors, the scripts were painstakingly debugged using Perl's command line debugger. The results of the investigation were that specific tag combinations in the legacy TMs were not recognized; by adding the necessary cases in the scripts, those tags were recognized and the scripts could finish executing properly. Once again, the results were verified by checking the scripts' log files to ensure proper execution and by checking a sample of the strings affected by the previously problematic tag combinations.

Removing Duplicates

Another aim of TM migration, besides providing a centralized and scalable access point, was to allow various TMs to be combined so that translation content would only have to be leveraged against a single, conglomerated TM, named the PG Doc TM. In the previous Trados-based method of translation, such TM recombination could not be as easily achieved as in WS; furthermore, the WS interface provided file metadata in a tabular format that made searching and organizing string entries very easy.

Recombination of TMs also produced undesirable duplicate entries. One consideration was that duplicate entries were redundant and took up memory. In turn, the performance of the workflow would be negatively impacted. A critical task was to remove duplicate entries in the legacy TMs before the legacy TMs could be added to the PG Doc TM.

Trados-based TMs stored translation pairs in the proprietary Trados format; this file format could only be opened by Trados and exported into an XML-formatted .tmx export file. In contrast, WS stored its data in an Oracle database, which was accessed and queried easily using SQL. An Oracle tool called SQL Developer was used to execute SQL queries that removed duplicates between TMs before combining them. Two desirable outcomes were achieved by reducing the number of duplicate entries before recombination of the TMs. First and foremost, redundant entries would be removed, thus solving the duplicate entry problem and ensuring that resources were used most efficiently. Secondly, the TMs remained separate, non-overlapping entities which could be reused in different combinations.

Further duplication problems remained that required the help of Language Specialists (LS), or translators, to resolve. While duplicate entries were simple source-translation pairings, multiple-translation entries were translations with the same sources but different translations. Determining the correct source-translation pairings necessitated the work of a translator. A Java servlet on the WS system was used to extract all of the multiple-translations for LS review. After LS reviewed the entries, any incorrect translations were removed from the legacy TMs via SQL Developer.

Deploying the TMs

Finally, metadata was added to the legacy TM entries. Each entry was given extra tagging that identified the entry's legacy name and type of content. In future, this information could be used to find specific entries from within the conglomerated PG Doc TM.

Completion of the TM Migration Process

A project Wikipedia page had been maintained during the trials and tribulations of migration work. Upon completing three successful legacy migrations, the wiki was refined one last time to document each of the common steps in those three successful migrations. The set of tools which had been created throughout the process was consolidated and clarified so that the next software engineer or developer would be able to modify those tools as necessary.

Conclusion

Through much trial and error, a vastly updated TM Migration process was born. This process was successful because each of the roadblocks, which had previously prevented legacy TMs from being converted for the WS system, had been resolved. Each of those resolutions was verified by examining the translation entries to ensure no corruptions were introduced. A well-defined process and set of tools that was established over the course of three difficult migrations now remains; each complete iteration of the process will add more polish and ensure that future migrations will be completed with much more efficiency and confidence.

Appendix A – TM Migration Process Wikipedia

General Migration Steps

Legend
= may or may not be necessary
= checkpoint

1. Export Legacy TM to TMX format.

If you can export directly from a Trados 5.5 TM into .tmx format, simply do that and proceed to Step 2.

However, if you must work with a Trados 7 TM, you will not be able to open the TM; instead, you will need to work with its .txt export, which must be provided for you. In order to convert from the Trados 7 .txt export to a Trados 5.5 .tmx export, you must first convert the Trados 7 .txt file format so that it can be imported into a blank Trados 5.5 TM, from which you can export the .tmx file. The steps are outlined as follows:

a) Get the provided Trados 7 .txt export. Then, use Notepad to open the exported .txt file, choose save-as, and verify that the default format selected is UTF-8. Close the file without saving. If the file is not UTF-8, you must save the file in UTF-8.

b) Use the Java native2ascii.exe utility to convert the file format.

Convert the previous UTF-8 .txt file into an ASCII encoding which contains all of the language's characters. For example, an ASCII encoding which fully supports French is Cp1252. Reference page with encodings is http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html. Native2ascii.exe can be found under C:\Program Files\java\<jdk version>\bin.

> **Conversion is a two-step process**
>
> i) Convert the UTF-8 file to an intermediate ASCII format first:
> native2ascii -encoding UTF-8 my_utf8_export.txt intermediate_file.txt
> ii) Convert the intermediate file to Cp1252:
> native2ascii -reverse -encoding Cp1252 intermediate_file.txt

my_Cp1252_file.txt

c) Get a new, empty TM from //depot/Localization/Projects/Mainline/TM/EmptyTMs/ for the appropriate language and unzip the contents to a directory.

d) Trados 7 export file may use a Unicode FONT which causes character corruption when trying to import the converted file (i.e. Cp1252 for French). This problem was seen in Cartesis Finance files. Use the "detect_ArialUnicodeMS.pl" script to scan the file for any Unicode fonts i.e. Arial Unicode MS. If any are found, the converted file is crawled and the number of Unicode font tags is tallied. Should any tags exist, the font definition at the beginning of the file must be modified so that the font is no longer Unicode i.e. edit 'Arial Unicode MS' to read 'Arial'

e) Import my_Cp1252_file.txt or the desired file into the new TM.

Sanity Check

Open the imported .txt file and find a corresponding translation unit in the TM. Code that appears within curly braces {<code>} should appear as a gray TAG icon within Trados. Ensure that special characters, like accents and apostrophes, appear normally and aren't corrupted as "?" characters in Trados.

f) Export the TM into .tmx format.

Checkpoint

After step e), special characters (like those with accents) should appear normally. If there are remnant ASCII representations of Unicode characters, such as \ufff, most likely there was an error in the conversion process. Check that your UTF-8 export from step a) is in Unicode UTF-8 format. If it is in ASCII, then the conversion steps corrupt the encoding.

2. Run the DITA Conversion Scripts on Legacy TM.

a) Search & Replace the .tmx files from 1f): in the case of Cartesis migration, the creationid and changeid contain invalid characters, the ampersand (&) symbol i.e. creationid="WH&P" so replace all instances of "WH&P" with "WHP" to proceed. In 'Search for' text box, enter: id="WH&P", type id="WHP" (incl. double quotes) in the 'Replace with' box and replace all of the strings.

b) Execute Perl migration scripts.

Migration Scripts & Documentation (sync everything in the LocScripts folder)

- //depot/Documentation/Tools/LocScripts/TradosTMXsplitType.pl

- //depot/Documentation/Tools/LocScripts/DITA_TradosTMX2Idiom_update.pl
- //depot/Documentation/Tools/LocScripts/DITA_TradosTM_update.doc

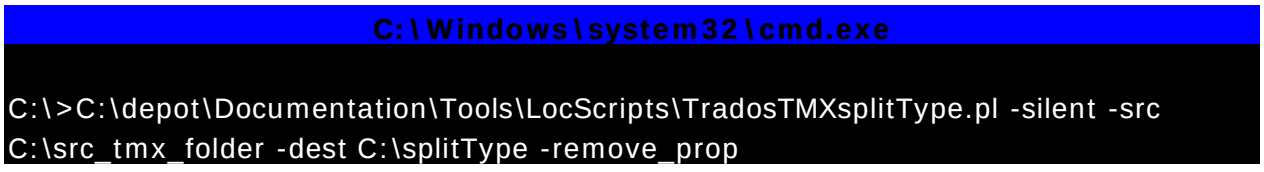
Verification Scripts & Documentation

- //depot/Localization/0_Idiom/dev/scripts/src/tm/CleanPhTmx.java
- //depot/Localization/0_Idiom/dev/scripts/src/tm/CleanTMX2.java
- \\vanpgsoftware\infobuilder\Team\Users\Trevar_EXIT_Tools\TM_Migration
- //depot/Localization/0_Idiom/Docs/TM_Migration_Steps.doc
- //depot/Localization/0_Idiom/dev/scripts/documentation/Instructions_cleanPhTmx.doc

i) execute TradosTMXsplitType.pl with the parameters described in DITA_TradosTM_update.doc

TradosTMXsplitType.pl reads the input .tmx files and creates another set of .tmx files that are divided into several categories; each original .tmx file may be split into multiple types, depending on the types of segments present.

See sample execution



ii) execute DITA_TradosTMX2Idiom_update.pl

Using the .tmx files split in the previous step, the script reads each translation segment to remove unnecessary tags.

When working on Cartesis content, the script terminates because it does not recognize some combination of tags. A fix is provided in this file, with which you overwrite the original, existing file in the \LocScripts\TMX folder:
//experiments/trchu/Trados2Idiom4DITA_FRAME.pm

See script instructions

Run this script up to two times:

a) After executing the script for the first time, unknown variables within the .tmx files may be inserted into the empty "unknown variable" text file which is passed as an argument. If the "unknown variable" text file remains empty, you do not need to run this script again; otherwise, proceed with the remaining steps

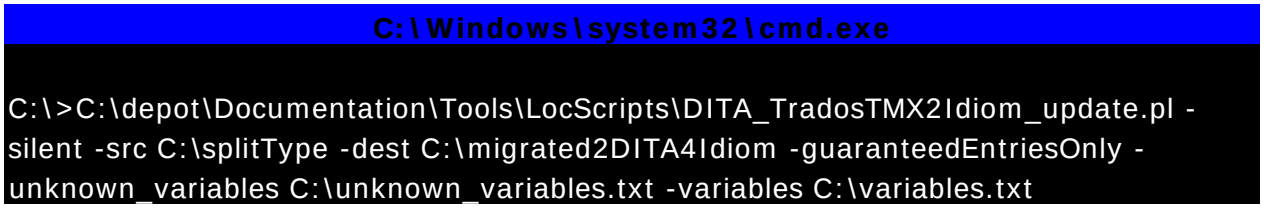
b) Separate each variable pair (e.g. BusinessObjects Enterprise=BusinessObjects Enterprise) by a line break
note this special case (remove ut tags): </ut>> some variable etc.<ut>=> some variable etc.

c) copy and paste all of the variable pairs to three sections within the variables.txt file: [en-gb], [en-us], and [*language you are working on*] and save this variables.txt (to be used again in the 2nd execution of this script)

d) clear the "unknown variable" text file (to be used again in the 2nd execution of this script)

e) execute the script a 2nd time using the updated variables.txt file as an argument; the "unknown variables" text file should be blank, and after the 2nd execution, should remain blank

See sample execution



iii) separate .tmx files

- place all migrated .tmx files with "_checkPH" in the filename into a separate "checkPH" folder
- place all migrated .tmx files with "_unvalid" in the filename into a separate "unvalid" folder
    - "unvalid" files are to be discarded and are no longer needed

> **Sanity Check**
> _checkPH files have "& l t ;!!!& g t ;" in their segments
> migrated files (non-checkPH and non-unvalid) should no longer have <ut> tags within them. Acceptable metadata include:
>
> - placeholder tags: <ph>
> - text representation of characters: "&" is "& a m p ;", "<" is "& l t ;", etc.
> - search for "ut>", "\cf6", "\cs6", ";:cs", ";:cns", ";:fn"
>   - if any are found, delete the segment containing the problem, or delete the whole .tmx file if too many segments are affected

iv) execute verification scripts

- run CleanPhTmx on the "checkPH" folder
    - remove the backup files with "_AC_NEW" appended to the filename
    - _checkPH files should no longer have "& l t ;!!!& g t ;" in their segments
- run CleanTMX2 on the "migrated2DITA4Idiom" and "checkPH" folders
    - if non-empty log files result, consult a language specialist

> **Checkpoint**
> All .tmx files to be migrated (checkPH and migrated2DITA4Idiom) only have placeholders and text representations of special characters.

3. Import Legacy TM and PG DOC TM into Dev WorldServer.

4. Remove Duplicates Within Legacy TM and PG DOC TM.

a) Install SQL Developer from: \\vanpgsoftware\software\Oracle\SQL_Developer

b) Find the Dev WS connection information for SQL Developer in this document on Perforce: //depot/Localization/0_Idiom/Docs/WS8_SystemConfiguration.doc

c) Find the Table IDs of your TMs. In Dev WS, go to Tools > Translation Memories > *Your TM*. In the Search Mode drop-down menu, choose Freeform SQL, type "x" and click the green arrow. Red error text should appear above the search box. Choose "View SQL..." and you will see the Table ID in the "from" clause, e.g. zTM1721_TRANSLATIONS

d) Connect to Dev World Server via SQL Developer and, for each of your Legacy and PG TMs, run this SQL query with the appropriate [ID] you found from step c):

```
DELETE
FROM zTM[ID]_TRANSLATIONS
WHERE rowid NOT IN
   (SELECT MIN(rowid)
    FROM zTM[ID]_TRANSLATIONS
    GROUP BY currentHash, targetHash
   );
```

5. Remove Duplicates Between Legacy TM and PG DOC TM.

a) Delete entries from Legacy TM where the sources and targets between the two TMs are identical.

```
DELETE
FROM zTM[LegacyID]_TRANSLATIONS
WHERE EXISTS
   (SELECT *
    FROM zTM[PGID]_TRANSLATIONS
    WHERE zTM[PGID]_TRANSLATIONS.currentHash =
zTM[LegacyID]_TRANSLATIONS.currentHash
    AND zTM[PGID]_TRANSLATIONS.targetHash =
zTM[LegacyID]_TRANSLATIONS.targetHash
   );
```

6. Resolve Multiple-Translation Entries Within the Legacy TM.

a) Run this query:

```
SELECT currentHash AS Legacy_Src_Hash, sourcevc AS Legacy_Source, targetvc
AS Legacy_Target, targetHash AS Legacy_Target_Hash
```

```
FROM zTM[LegacyID]_TRANSLATIONS
WHERE currentHash IN
   (SELECT currentHash
    FROM zTM[LegacyID]_TRANSLATIONS
    GROUP BY currentHash
    HAVING count(currentHash) > 1
    )
ORDER BY Legacy_Source;
```

b) Right-click in the Results of the query, choose Export Data -> XLS and save the .xls file to disk.

c) Hide the two hash columns and send this .xls to the LS for review.

i) LS will review the rows of data and remove all valid entries. Any entries that the LS leaves in the .xls spreadsheet are invalid TM entries and thus will be removed from the TM!

d) If any entries need to be removed, LS will return the .xls file. Delete any rows in the .xls that are *not* marked for removal. Create a temporary removal (zTM_REMOVE) table in SQL Developer and import the hash columns from the .xls file into a two-column table. Name the columns according to the same convention as the TM tables, i.e. currentHash and targetHash. This table now contains two columns of all the hash entries to be deleted from the Legacy TM. Remove TM entries from the Legacy TM by deleting the intersection of the following two tables:

e) Run this query:

```
DELETE
FROM zTM[LegacyID]_TRANSLATIONS
WHERE EXISTS
   (SELECT *
    FROM zTM_REMOVE
    WHERE zTM_REMOVE.currentHash = zTM[LegacyID]_TRANSLATIONS.currentHash
    AND zTM_REMOVE.targetHash = zTM[LegacyID]_TRANSLATIONS.targetHash
   );
```

7. Resolve Multiple-Translation Entries Between the Legacy TM and the PG DOC TM.

a) Generate a list of entries where the source between the two TMs are identical, ignoring placeholders, i.e. "Hello,{1} World" and "Hello, World" are interpreted to be

identical. On the WS home page, choose [TM Servlet] - TM Comparison. Select the Legacy TM and PG TM in the drop-down menus, specify the HTML's output folder, and check "Ignore Placeholders."

b) When the servlet completes, open the HTML file that was generated, ensuring that the page's encoding is set to Unicode (UTF-8), then copy & paste its contents in an .xls file, preserving the column and row structure. After pasting the contents into Excel, select all the cells, go to Format > Cells.

i) In the Alignment tab, uncheck "Wrap text."
ii) In the Border tab, select "None" under the Presets section.

Finally, in Column E, add a header which reads "Remove From Legacy?" and send this .xls sheet to the LS.

c) LS will review the .xls file and mark each entry to be either removed from the Legacy TM or to be preserved.

d) If you have strings with a single quote character ('), as is often the case in French, you will need to add another single quote character in front of each occurance in the .xls sheet to act as an escape character. Then, each double occurance of single quotes ('') will show up as one when imported into the database. Note that double single-quotes ('') are not the same as a double-quote (").

e) Remove invalid legacy multiple-translation entries from Legacy TM; copy the rows marked for removal from step 7b to another *removal* .xls file. Create a two-column, temporary removal (zTM_REMOVE) table in SQL Developer and import the Legacy source and target columns from the *removal* .xls file. Remove Legacy TM entries similar to the previous step. Note that due to the nature of the comparison servlet, the TM entries are matched in the SQL by their strings instead of their hashes (as was the case in the previous step).

Sanity Check

After importing the .xls file into a database table, check that the number of rows you imported corresponds to the number of rows in the .xls sheet. Furthermore, verify that the number of distinct rows in the removal table correspond to the number of rows that you will be removing in the next step.

f) Run this query:

```
DELETE
FROM zTM[LegacyID]_TRANSLATIONS
WHERE EXISTS
   (SELECT *
    FROM zTM_REMOVE
    WHERE zTM_REMOVE.LegacySource = zTM[LegacyID]_TRANSLATIONS.SOURCEVC
    AND zTM_REMOVE.LegacyTarget = zTM[LegacyID]_TRANSLATIONS.TARGETVC
   );
```

Sanity Check

The number of rows in step f) should either correspond to the number in step e), or the number that is returned by this query:

```
SELECT DISTINCT *
FROM zTM_REMOVE
```

8. Create a Set of Valid Multiple-Translation Entries with Additional Tags:

In order to preserve Legacy translations that do not have placeholder tags, we add placeholders so that the Legacy content will be matched.

See explanation

Before Placeholder Tagging

Legacy source	Legacy target	PG Doc source	PG Doc target
AB{1}	CD{1}	AB{3}	EF{3}

Translation of "AB" partially matches "CD{1}" and "EF{3}" but translation of "AB{3}" results only in "EF{3}" in these TMs.

After Placeholder Tagging

Legacy source	Legacy target	PG Doc source	PG Doc target

AB{1}	CD{1}	AB{3}	EF{3}
AB{3}	CD{3}		

Translation of "AB{3}" now results in "EF{3}" and "CD{3}" in these TMs, thus preserving the Legacy translation.

We achieve this by performing the following steps:

1. Take source from PG Doc TM.
2. Take target from Legacy TM.
3. Remove all placeholder tags from the Legacy target
4. Append all the placeholder tags from PG Doc source to the Legacy target
5. Add this set of translations to the Legacy TM.

a) Create an .xls of all the rows in step 7b that are to be preserved.

b) Execute Legacy_Target_Tagging.java -src [column index] -trg [column index] [.xls file]

- the source column index (starting from 0) is the PG Doc source column
- the target column index (starting from 0) is the Legacy target column

c) Import the resulting Legacy-tagged .tmx file into the Legacy TM through the WorldServer interface.

d) Through SQL Developer, remove duplicates within the Legacy TM.

```
DELETE
FROM zTM[LegacyID]_TRANSLATIONS
WHERE rowid NOT IN
   (SELECT MIN(rowid)
    FROM zTM[LegacyID]_TRANSLATIONS
    GROUP BY currentHash, targetHash
   );
```

9. Add Properties Tags to All Entries.

a) Export the Legacy TM into a UTF-8 .tmx file.

b) Execute PL_PRN_Tagging.java [source folder] [output folder]

Sanity Check

Ensure that the new <prop..>..</prop> tags are inserted correctly between the <tu> and <tuv> tags.

10. Add Legacy TM to PG DOC USER Migrated TM on Production WS.

a) Import the .tmx from step 9b into the PG DOC USER xx-XX TM on WorldServer.