Translation Memory Administration Guide

SDL WorldServer 2011

SDL WorldServer™

Version 10.0

SDL Language Technologies | 69 Hickory Drive, Waltham MA 02451 USA

January 2011

Contents

Preface

About the SDL WorldServer Translation Memory Administration Guide

Welcome to SDL Enterprise Technologies and WorldServer™. SDL WorldServer is a leading globalization software solution. It extends your existing enterprise content management (ECM) systems, software configuration management (SCM) systems, file systems, and databases with world-class globalization capabilities.

Scope

The *SDL WorldServer Translation Memory Administrator Guide* is a document for advanced administrators who need to become familiar with the details of translation memory. It describes translation memory leveraging, how translation memory matches are scored, TM groups, segment repair, segment identifiers (SID), TM properties, alignment, sentence breaking, and more. It is the essential reference for sophisticated use of translation memories.

SDL WorldServer Documentation and Resources

The SDL WorldServer documentation set includes the following deliverables.

Note: Unless noted otherwise, find these resources from the **WorldServer Product Documentation** link on the WorldServer Home page.

Documentation Deliverable	Description
Administrator Guide	Information on setting up and administering WorldServer in your environment.
Administrator Tutorial	Leads an administrator through the process of performing an initial configuration of WorldServer.
Browser Workbench User Guide	Information on embedded translation tool for brief reviews or edits.
Glossary	Defines terms specific to WorldServer.
Install and Upgrade Guide	Instructions for installing or upgrading WorldServer and its components in your environment.
Translation Memory Mode Concepts Guide	Information for project managers that need to understand the Studio-aligned translation memory mode.
User Guide	Information for general WorldServer users: project managers, translators, and reviewers.

Documentation Deliverable	Description
Using SDL WorldServer with SDL Studio 2009	Information for translators, reviewers, and project managers that use SDL Studio 2009.
Software Development Kit (SDK) User Guide	Reference guide for APIs that extend WorldServer functionality and integrate WorldServer with other custom or third-party applications.
More Resources in the `doc` directory at the top level of your installed WorldServer distribution. The full set of PDF files are found in this location.	
Release Notes	Late-breaking information about known issues and issues fixed in this release. (Found in the `en` folder. Most recent updates may be found on the `FTP` site.)
SDK Web Services Developer Guide	Reference guide for adding Web Services to WorldServer. (Found in the `Additional Documentation` folder.)
Supported Configurations Guide	Lists supported and unsupported machine and operating system combinations for WorldServer. It also covers the history and evolution of supported configurations. (Found in the `en` folder.)
Translation Memory Administrator Guide	Reference guide for understanding and configuring the WorldServer translation memory. (Found in the `en\Additional Documentation` folder.)
More Resources available from the **Help** link in the WorldServer window, and the **More Info** link on individual WorldServer pages.	
Online Help TransPort Online Help	Provides context-sensitive help for each page in WorldServer or WorldServer Transport.

Chapter 1

Translation Memory Administration Overview

Topics:

The WorldServer Translation Memory (TM) provides customers the ability to leverage earlier translation efforts. This results in minimizing costs associated with translating new and modified content. The WorldServer TM technology has undergone enhancements geared at optimizing historical translation content for reuse in current translation efforts. SDL has employed technological advancements, such as the segment repair options, and split/merge technology, yielding substantial savings for WorldServer customers. WorldServer has repeatedly demonstrated its ability to bolster leverage of third-party data that has been migrated into the WorldServer environment.

The WorldServer TM exposes many controls and options, each of which affects the overall system in varying ways. The ability to use these technologies effectively requires you to understand both the WorldServer TM technology and the administration options. Some controls affect performance and quality, some affect leverage, and others affect other operations. Understanding each option and its impact helps you tune the WorldServer TM to meet your needs and requirements.

This guide provides a reference guide for understanding and configuring the WorldServer TM.

WorldServer TM Leverage Overview

The following section provides a very high level overview of the WorldServer translation memory leverage process. It is intended only to jump start your understanding of the TM leverages process, and is geared toward the default behavior within WorldServer. See following chapters to gain a greater understanding of the technology and the large number of options that are available to you.

Running the TM Leverage Process

Segmentation is the process through which the content within a source asset is broken into translatable chunks called segments. In the context of a typical WorldServer project, segmentation is triggered by the inclusion of the `Segment Asset` automatic action. When working with assets directly from the WorldServer Explorer, segmentation is triggered the first time the asset has been opened for translation. It will also be triggered the first time the asset is reopened following changes to the assets.

The translation memory leverage process is generally associated with the segmentation process. As segments are being created, they are passed to the translation memory engine, which attempts to find the best match for the segment. In order for the TM leverage process to be run during the segmentation process, a TM or TM group must be associated with the target folder. If there is no TM or TM group assigned, then leverage will not occur during segmentation.

Finding the Best Available Matches

During the leverage process, the target segment is auto populated with a translation from the TM (or TM group) for the following type of matches ranked in the following order:

1. **ICE/SPICE matches** – In-Context Exact matches and Segment-Preferred ICE respectively. These matches are exact matches that include some additional usage context that makes them guaranteed matches. By default, SPICE matches are preferred over ICE matches. However, this is configurable.
2. **Exact matches** – The content is exactly the same as the TM match. This includes everything, including whitespace.
3. **100% matches** – These are matches that score 100%. These include exact matches, but may include other matches with differences that are not penalized. For example, if there is a whitespace difference, but there is not a penalty defined for whitespace differences, then the match may result in a 100% match.
4. **100% Repaired Fuzzy** – These are matches that were not 100% matches initially, but were repairable up to 100% status through the use of WorldServer and custom repairs.
5. **Auto-translated segments** (not the same as machine translation) – Segments that fit a certain pattern as defined by the auto translation module (whether standard WorldServer module or a custom module) will be auto translated and awarded a 100% score.

Matches below 100% are not auto-populated during the leverage process. The leverage process will record the score for the best fuzzy match found so that it will be reflected in the scoping analysis. The pre-translation operation can be used to populate untranslated segments with the best available fuzzy match provided that the best available fuzzy match meets or exceeds the minimal score requirement (which is 75% by default.)

There are additional match ranking rules that are used to for breaking ties for multiple match candidates of the same leverage type or leverage level. These are summarized below:

1. **SPICE matches** – There is no tie breaker since there can be only one SPICE match in the TM for a segment. If a TM group is used, the SPICE match for the highest priority TM is used, provided that another 100% or better match was not found in a higher priority TM.
2. **ICE matches** There may be multiple ICE matches in the system for a segment. There are tie breaker rules for determining the best available ICE match. The tie breaker rules are:

   a. ICE matches with a full usage context are preferred over partial usage context matches.
   b. ICE matches with full mapped metadata match are preferred over partial/no mapped metadata matches. (A partial match of mapped metadata is not preferred over no mapped metadata match.) In other words, if two TM entries

for “Hello” exist as “A” and “B”, but only “B” has a matching business area in comparison to the source folder, “B” will be the corresponding TM match.

   c. Matches associated with the asset being translated are preferred over matches associated with a different asset.
   d. For ICE matches associated with an asset where the segment being translated is a repeated segment within the asset, the TM will prefer the exact whose position matches closest to the segment being translated.
   e. As a final rank condition, and all else being equal, the most recent entry will take precedence.

3. **Exact matches** – There may be multiple exact matches in the system for a segment. There are tie breaker rules for determining the best available exact match. The ranking rules for exact matches are similar to those of the ICE ranking process:

   a. Exact matches with full mapped metadata match are preferred over partial/no mapped metadata matches. (A partial match of mapped metadata is not preferred over no mapped metadata match.) In other words, if two TM entries for “Hello” exist as “A” and “B”, but only “B” has a matching business area in comparison to the source folder, “B” will be the corresponding TM match.
   b. Matches associated with the asset being translated are preferred over matches associated with a different asset.
   c. For exact matches associated with an asset where the segment being translated is a repeated segment within the asset, the TM will prefer the exact whose position matches closest to the segment being translated.
   d. As a final rank condition, and all else being equal, the most recent entry will take precedence.

4. There are not enforced rankings for other 100% or repaired 100% matches.

Note: The next TM within a TM group is only searched when the current TM being leveraged does not produce a match that scores a 100% or better match. Stated another way, a repaired 100% match from a higher priority TM will be chosen over an ICE match from a lesser priority TM.

TM Match Penalties

It is important to understand that TM match statuses and the TM match score can be affected by two types of penalties—content and leverage level penalties.

Content level penalties are the most easily understood because they are the penalties that are related to content differences between what is sought versus what is found. For instance, if a TM match has an extra word or has different punctuation than the segment being leveraged, the score will reflect the content differences.

However, leverage level penalties are penalties that affect the final match based on external requirements. For example, the maximum target length penalty is assessed not because of content differences, but because the target in the match happens to be longer than an externally imposed restriction. Leverage level penalties can reduce SPICE, ICE and exact matches to high fuzzy matches.

Viewing TM Matches for Segments

Available matches for a segment can be seen using the Browser Workbench’s segment match lookup tool or using the standard TM search tools. The results of the browser workbench tool will filter out matches the score less than the browser workbench configured minimal score for viewing. These same matches will be available to Desktop Workbench users within the translation kits depending on the configured match export level. (By default, extra matches are not exported for 100% and ICE matched segments.) In addition, matches with the same translation will be collapsed into a single displayed entry. The TM search tool can be used to see all available matches regardless of score or common translations. Another important distinction between the two tools is that the TM search tool does not invoke the repair process and the results are not impacted by leverage level penalties since the search are done outside the context of a specific segment or asset.

Note: The search process for finding fuzzy matches during the leverage process and in the Browser Workbench has been optimized to balance performance and quality requirements. The fuzzy lookup process does well to find most of the best matches while requiring the least amount of data to be retrieved from the database. However, the process is not always guaranteed to find the best match. At the sacrifice of system performance, it is possible to tweak the system to improve the probability of finding the best fuzzy match. In most cases, the default configuration does in fact provide the most optimal results.

This guide will delve deeper into the topics highlighted in this overview, and will cover other advanced topics integral to understanding the capabilities of the WorldServer translation memory system.

Definitions

The following definitions describe the terminology used throughout this document.

100% Match

A *100% match* is a translation memory match with a score of 100%.

Additional Information

Note: Prior to WorldServer 8.0, using a different type of space character from that of the TM entry prevented it from being selected as an exact match. This difference was not penalized during the scoring. As of WorldServer 8.0, this is no longer true. Whitespace differences are now treated as punctuation differences, and the punctuation penalty will be assessed against the match score.

For information about match scoring, see *Match Scoring Configuration* on page 109. For information about segment repair, see *Segment Repair Technology* on page 97.

Asset

An *asset* is a unit of information that you are working with or translating, such as a text document or a database record.

Asset Interface System (AIS)

The *Asset Interface System*, or AIS, is the framework WorldServer uses to access content whether that content is in a content management system, a file system, a database, or any other format.

AIS Path

The *AIS path* is the location of an asset managed by the WorldServer Asset Interface System (AIS). The AIS path contains a complete hierarchy of folders starting from the mount name and ending with the asset.

AIS Context

The *AIS context* refers to the environment in which the segment is being translated. The asset in which the segment exists defines this environment. The AIS Context is based on the AIS path, including the name of the containing asset. Thus, it is unique for a given asset.

Auto-translated Segment

Auto-translated segments are segments that are translated by the system automatically. The text associated with the target is identical to the source text. When the segment is auto-translated, the source text is used to populate the translation text.

Auto-translation

Auto-translation is the process through which translations are generated automatically from source content that adheres to established content patterns.

Exact Match

An *exact match* contains source text that is completely identical to the lookup text from the asset at the moment it comes out of the translation memory. These matches receive a score of 100%.

Filter

A *filter* is a component that performs the segmentation of *assets*, extracting translatable content for a particular file format.

Fuzzy Match

A *fuzzy match* is a translation match with a score of less than 100%.

In Context Exact (ICE) Match

Most translation memory (TM) systems employ a string comparison algorithm to gauge the appropriateness of a translation stored within the TM database. Scores are awarded based on how closely the two strings match. A score of 100% typically indicates that an exact match has been found. While some systems stop at this level of matching, WorldServer offers an improved category called *In Context Exact (ICE) match*.

To be characterized as an *ICE match*, a match candidate must satisfy the basic criteria for ICE matching — the source text must be an exact match, and must share a common usage context with the translation memory entry providing the match. These criteria can yield more than one qualifying match candidate. Ultimately, the candidate that most closely matches the lookup segment will be chosen based on the ranking strategy. The ordered list represents additional criteria that are taken into consideration to break the tie (or rather to deduce the most appropriate candidate.)

For instance, candidates that share a complete usage context (defined as being used between the same segments as the source/lookup segment) will be chosen over those that share only a partial usage context with the source or lookup segment. The ICE matching process consists of these basic steps:

1. Find exact match candidates.
2. Evaluate for ICE condition.
3. Rank candidates.

When a segment has an in-context exact match, the target segment has a purple bar to its left. When a locking workflow is used, there is also a lock icon that appears, and the translator cannot modify that segment.

Leveragability

Leveragability describes the extent to which the content being translated can make use of the data in the applied TM. Leveragability is measured by the results from the TM scoping process. For a given document, leveragability shows what percentage of the document falls into each leverage level.

Leverage Level

Leverage level refers to the extent to which content can be translated against the applied TM. The higher the leverage level, the more the TM content is reusable. Leverage levels are generally decomposed into the following levels: ICE/SPICE, Exact/100%, and interval fuzzy ranges.

Leveraging (or Match Lookup)

Leveraging refers to the process during which content is matched against TM entries within a translation memory. During this leverage process, a specific TM is applied to the content of the selected assets. Using the historical translation data (as represented by the TM entries) reduces the effort needed to translate new or modified content.

Lookup Text

Lookup text refers to the segment text from the source asset for which a TM match is to be sought. For example, if you are translating from English to French, "I don't know" is the lookup text.

Path Normalization

Path normalization allows the customer to determine how the TM defines the TM AIS context for TM entries. By default, each asset has its own TM AIS context, and thus, has its own set of TM entries when it is translated. The path normalization process can be overridden to force physically different assets to have the same AIS context, and thus, share a common set of TM entries.

Placeholder

Translatable content is often mixed with formatting information. This formatting information is important, but does not always represent information that should be translated. Instead of exposing this information explicitly, this information is represented by content tags called *placeholders* embedded into the segment. For example, consider the following HTML text:

```
“This <I>content</I> contains <B>formatting</B>.”
```

WorldServer represents the text as follows, with placeholders replacing the formatting:

```
“This {1}content{2} contains {3}formatting{4}.”
```

The placeholders are shown to convey the position of the embedded formatting information without presenting the information for translation.

Repeated (Segment)

A *repeated segment* is the first occurrence of a duplicated segment, either within the current asset or within another asset contained in a specified collection of assets (such as within a project.) All other occurrences are considered repetition segments.

Repetition (Segment)

A *repetition segment* is a subsequent occurrence of a duplicated segment within an asset contained in a specified collection of assets (such as within a project.) The first occurrence of a duplicate segment is considered the repeated segment.

Scoping

Scoping is the process of comparing content to be translated with past translations to assess the remaining translation cost and effort.

Segmentation

Segmentation is the process through which asset content is parsed and exposed as translatable chunks or segments. The size of the segment depends on segmentation rules. In practice, a segment can represent a paragraph, a sentence or even a sentence fragment. Segments typically are not single words, though single word segments can result from the segmentation process.

In general, SDL encourages its customers to define their segmentation strategies to coincide with a sentence-based approach, as supported by default WorldServer filter configurations.

Segment

A *segment* is the base unit of translation. A segment generally is one sentence long, but segmentation is controlled by the *filter*.

Segment Identifier (SID)

A segment identifier, or *SID*, is a label that defines the usage context in which a segment is to be translated. Instead of basing the usage context on surrounding segments, the usage context can be explicitly defined by applying a SID to the segment. SID support must be enabled for WorldServer to find *SPICE matches.*

Segment Repair

Segment repair is the process where WorldServer applies defined algorithms for the goal of correcting differences between the lookup text and the TM entry. The objective is to improve the match leverage level by applying high quality heuristics.

Shingles

Shingles is a term used to refer to word runs or a series of sequential words taken from reference text. The translation memory fuzzy match process uses shingles to select match candidates. Instead of determining candidates based solely on having common words, the process finds match candidates based on having common word runs.

SID Preferred ICE (SPICE) Match

SPICE match stands for “segment preferred in-context exact match.” It is an *ICE match* that is found by using the *SID* value to define the usage context. The combination of the SID and the text of the segment can resolve only to a single translation for a given source and target locale combination. SID support must be enabled for WorldServer to find SPICE matches. By default, SID support is enabled in WorldServer, though it cannot be used without a SID-based filter being installed.

Source Text

Source text refers to the text within the translation memory entry or the *segmented asset* that corresponds to the original (source) language, which is the language being translated. During the leverage process, the source text in the translation memory is compared to the lookup text from the asset in order to find a match.

Stemming

Stemming is the process of reducing morphological variants of words (for example, `cats` and `cat`) to a common root form or stem (for example, `cat`) that can be used to identify all variants of the word. Stems can be used to find matching entries containing word variations. Stemming can generate a larger number of match hits.

Target Text

Target text is one half of a translation memory entry. It is the translation of the *source text* for a particular language.

Translation Memory

A *translation memory*(“TM”) is a database of segments and their translations.

TM AIS Context

The *TM AIS context* is the asset context to which translations are associated. Typically, this is the same as the AIS context, and can be used interchangeably. Whereas the AIS context allows for a specific asset to be located, the TM AIS context allows the *translations* for a specific asset to be located. The TM AIS context is unique for any asset by default. However, customers can alter how the TM determines the AIS context by overriding the TM path normalization process.

TM Entry

The *TM entry* is a translation pair stored in the translation memory that maps source text to translation (or target) text. It is specific for a given source and target locale pair, and is generally associated with the asset whose translation produced this translation pair. In effect, a TM entry represents a previous translation, which can be reused later.

WorldServer supports the migration of translation memories from third-party applications, such as TRADOS. If the TM entry originated outside of WorldServer, it does not have any associated asset (AIS context) information stored with it. The following example shows a TM Entry in which the source locale is English and the target locale is French:

```
Source Text: “I do not know.”
```

```
Target Text: “Je ne sais pas.”
```

Usage Context

The *usage context* refers to the circumstances that influence how WorldServer derives the appropriate translation for content – by considering the text surrounding the text to be translated. Typically, the usage context is defined in conjunction with surrounding content, which provides insight into the meaning of the segment to be translated. During translation, a translator instinctively uses the surrounding segments to infer the proper meaning and translation for a given segment. Similarly, WorldServer considers the usage context to find an ICE match for a given segment.

Chapter 2

Translation Memory Management

Topics:

The WorldServer translation memory technology is vast and powerful, and provides you with substantial financial benefits. However, to really tap into these benefits requires that your translation memories are properly configured and well maintained. Translation memories will need maintenance at various points during their lifetime. You may need to remove legacy data, correct mistakes, import new data, and export data in order to create new translation memories. Additionally, you may find it necessary to reorganize your translation memory based on new business objectives or to capitalize on new WorldServer features. Toward this end, SDL has provided an assortment of translation memory tools to assist you in managing your translation memories and translation memory groups.

The translation memory tools provided by WorldServer support the following capabilities:

- create, delete, and rename translation memories and translation memory groups
- import TMX data into translation memories
- export TMX data from translation memories
- delete translation memory entries
- edit and update translation memory entries
- restore history entries
- search translation memories and translation memory groups
- perform search and replace on translation memory content
- define custom attributes for translation memories and translation memory translations or entries

This chapter focuses on the concepts and usage of various TM management operations. Please refer to the *WorldServer User Guide* and online help for more details on performing the discussed operations.

Basic Translation Memory Usage

You can create and delete translation memories, and you can create custom attributes that enable you to extend the out-of-the-box functionality. To manage you translation memories, WorldServer provides tools for:

- Importing TMs
- Exporting TMs
- Deleting TM entries
- Updating TM entries
- Searching TMs
- Searching and Replacing TMs

Translation Memory Creation

WorldServer allows you to create as many translation memories as you need to support your business needs. The creation of a new translation memory means that you provide WorldServer the information it needs to define a new container for translation entries. Minimally, you will need to provide a name for the translation memory. Additionally, you can set the values for any and all of the configured custom translation memory attributes. Once you have created a new TM, you can start using it.

It is important to realize that you do not need to define which source-to-target language combinations are to be used with this translation memory. Any translation memory can be used to store translations from any source-to-target combination. You will ultimately control which translations are stored in the translation memory by the AIS paths to which the translation memory is assigned, and by the TMX files you choose to import into the translation memory instance.

Translation memories must be assigned to AIS paths in order to be populated during translation operations. In short, the process involves the following steps:

1. Create a new translation memory.
2. Set the translation memory property on each target AIS path containing the assets that you want to be leveraged against the translation memory.
3. Translate assets. The associated translation memory will be the translation memory into which new translations from these assets will go.

Thus, the translation memory will contain entries for source-to-target combinations based on the assets that were translated using the translation memory.

The above applies to translation groups as well.

Translation Memory Deletion

Translation memories can be deleted. Perhaps a particular translation memory is no longer required. If this is the case, it may be desirable to delete the translation memory. Translation memories consume space in the database. If the translation memory is not to be used further, you should delete it. It would be wise to make sure there is a backup of the translation memory before deleting it, in the event it is discovered that the translation memory is being or will be used.

If the objective is to recreate the translation memory, then you should consider other options outside of deleting the translation memory. When you delete translation memory, all AIS paths that have been associated with the translation memory will be unset. This means that when translation operations are initiated on the assets underneath these AIS paths, they will not have a defined translation memory to be leveraged against, nor will there be a translation memory to accept the new translations.

Consider the following alternatives to deleting the translation memory for the following scenarios:

- *Incorrect data was imported into the translation memory:* Each import job is given a job ID. You can use this job ID to perform a query-based deletion from the translation memory. The job ID is displayed in the log of the import process. If you performed a foreground based import and did not capture the import job ID, you can query to locate one of the records from the job, and grab the job ID from it. If you performed a background import job, you can view the log from the background process via the background job monitoring facilities. (Please see the *WorldServer User Guide* for additional details.)
- *An import of a TMX file failed:* If you want to keep the data, simply repeat the import process. WorldServer will not create duplicates of each of the TMX records.
- *Recreate the translation memory with same name:* In this case, you should use the Purge TM Data tool, which will empty your translation memory, but will not affect the AIS property assignments for the translation memory.
- *You want to create a new translation memory with the name of a current translation memory:* Instead of deleting the current translation memory, simply rename it. This will maintain all existing AIS path associations. Whether the old translation memory is renamed or deleted, you will need to manually associate the new translation memory with the desired AIS paths.

In short, consider deleting the translation memory only when you really want to destroy the translation memory, all of its data, and all associations.

It is important to note that you cannot delete translation memories that are used by TM groups. You will need to manually remove the translation memory from each TM group that references it. Fortunately, WorldServer makes it easy to identify which TM groups are referencing a translation memory. When you attempt to delete a translation memory in WorldServer, the delete screen will give you a list of all TM groups referencing each of the selected translation memories. You will also have active links for jumping to the configuration page for each of the TM groups.

Custom Attributes

Defining Custom Attributes

WorldServer allows you to create custom attributes for both translation memories and translation entries. Custom attributes allow you to define fields that further aid you in describing or defining specific instances of objects. For instance, you can create a custom translation memory attribute that identifies what company a specific translation memory is associated with, or you can create a translation entry attribute that identifies the product the translation is associated with. (See the online documentation for creating custom attributes.)

After you define custom attributes, you can assign values to them for each TM or TM Entry. You can set the translation memory attributes when you create or edit the properties for translation memory. Similarly, translation entry attribute values can be set within the translation entry editor. Alternatively, values for translation entries can be pulled from the values of AIS properties of the associated asset for AIS properties that have been mapped to the translation entry attributes. (For more information, refer to *TM Entry Attributes* on page 87.)

Property values can also be set using custom processes such as automatic actions.

Using Custom Attributes

How custom attributes are used is completely up to you and your needs. Most attributes and their values do not affect the normal operations of WorldServer. Nonetheless, there are exceptions: for instance, translation entry attributes can be used to influence the match ranking process. (For more information, please refer to *TM Matching Based on Metadata* on page 93.)

Beyond the standard WorldServer supported operations, attributes can be used to facilitate a number of custom processes. Consider the following scenarios.

- *Custom TM Group Member Validation* – perhaps you want to prevent a translation memory group from being defined with translation memories from different associated companies. You might create a custom process that periodically validates translation memory groups and removes all translation memories from it that do not match the company that is associated with the update translation memory within the translation memory group.
- Export Translation Entries with a Specific Attribute Value – perhaps you want to create a new translation memory with only a subset of data based on attribute values.

The use of attributes does not have to lead to any automated processes or special processes. They can simply be used to provide additional background information.

Translation Entry Management Tools

The WorldServer contains a collection of online tools to help you manage the content of your translation memories. These tools allow you to import, export, find, delete and modify translation entries stored within your translation memory. Many of these tools are restricted to translation memories, and are not available for translation memory groups. The following table shows the operations supported by the management tools, and which translation memory entities support these operations in the WorldServer UI.

Operation	**Supported for TMs**	**Supported for TM Groups**
Imports	Yes	No
Exports	Yes	Yes (Query Based Only)
Entry Deletion	Yes	No
Entry Update/Restore History	Yes	Yes (currently)
Search	Yes	Yes
Search and Replace	Yes	Yes

Many of the operations are not made available to translation memory groups via the WorldServer UI in order to provide a level of protection to the shared translation memories. Use of a translation memory group should not give full rights to the included or referenced translation memories.[1]

The remainder of this section will discuss key notes on each of these supported operations.

TMX Import Tools

WorldServer provides tools for allowing you to import TMX data into an existing WorldServer translation memory. The TMX data may have been exported from an existing WorldServer environment or it may have been generated from a third-party application. WorldServer will support the import of either; however, TMX data originating from a third-party application will not contain any of the usage context or AIS context information that is generated during the WorldServer translation process. This generally has two implications:

- Basic ICE matches will not be possible. SPICE matches may be possible, provided that segment identifier (SID) information has been encoded into the TMX file.
- Without AIS contextual information, WorldServer cannot associate the entry to validate WorldServer assets. This means that you cannot display the associated WorldServer asset when using the search tools.

The import process has been updated to allow the customer to re-import a third-party TMX file multiple times without generating an entirely new set of TM entries. *The process is based on the name of the TMX file being imported (including*

[1] The SDK provides support for some options that are not supported in the WorldServer UI. Customers should thoughtfully consider the implication all functionality exposed to users via SDK based customizations.

the path), and the language mappings that have been selected. If a newer version of the TMX file has been created with additional or modified entries, only the new or modified entries will be imported into the TM.

Note: If a TMX file is renamed and is re-imported using the new name, the import process will generate a new set of TM entries for the TMX file. In essence, the import process will treat the TMX file as the associated asset for the created entries. Additionally, if you re-import the TMX file, and map the languages differently, it will result in additional entries.

Background and Foreground Processing

The TMX import tool is an important tool for transferring or migrating TMX data. The process can also be resource intensive, particularly if performed in the foreground. WorldServer now supports the ability to run imports in the background. Administrators can configure user rights that will allow both foreground and background processing of import jobs. Administrators can also prevent foreground processing of import jobs all together.

Running too many import jobs in the foreground can bring down the WorldServer environment. For this reason, SDL recommends that running import jobs in the foreground be restricted. Running jobs in the background will provide all of the same log information, and allows for jobs to be queued up for execution in a manner that is managed by allocated system resources. Import jobs can be prioritized and canceled. See the section "Working with Background Jobs" in the *WorldServer User Guide* for more information.

TMX Export Tools

The TMX export tools allow you to export translational data from translation memories. This data is exported in TMX format, and can be imported into another WorldServer translation memory or any other third-party translation memory system that supports the TMX format. Exporting translation memory content may be useful for transferring context to another environment, another translation memory within the same environment, or simply to provide a backup of the translation content.

WorldServer allows users to define query-based exports in addition to the standard TM attribute filter-based exports. This extended functionality allows for more specialized collections of content to be exported. For instance, you could even export the content associated with a specific asset, or simply all the entries containing a specific word.

Search Tool

One of the most popular management tools in the WorldServer collection is the translation memory search tool. Often you may be interested in viewing content in a specific translation memory or translation memory group. The search tool is a very important translation and leverage validation tool. You may use the tool find matches that you expect or suspect are in the translation memory. Often this tool helps you fully understand and validate the results of the translation leverage process. Additionally the search tool provides a means to define result sets that can identify records that you may want to delete, export or update.

There are several search modes. It is important to understand the differences between these search options.

- **Standard**. This exhaustive search finds all matches in the translation memory that exactly contain the search text (ignoring whitespaces). It supports searches for words, partial words (using wildcards), numbers, and non-alphanumeric characters. Note that searches using this mode may not return every occurrence in the translation memory of the specified search term, because the `Maximum Number of Hits` value limits the number of entries returned.
- **Segment Leverage**. This search mode (which is similar to what in releases previous to WorldServer 9.0 was called Fuzzy Search mode) shingles the search text to look for matching terms. If the search text is `fox in socks on Knox on box`, potential terms that can match include `fox in socks`, `socks on Knox` and `socks on Knox on box`. This search mode only supports searches for whole words. It takes advantage of stemming, if set up, and returns results equivalent to doing a translation memory lookup from a segment in the Browser Workbench. Segment Leverage searches are only guaranteed to find matches if enough context is supplied (typically 3 words). For example, searching for a single word only finds matches in short segments but does not exhaustively find matches, because not enough word context was provided.

  Wildcard search ("`*`") only works in **Standard** search. It does not work in **Segment Leverage** search.

- **Concordance**. The concordance search tool lets you search translation memory and, optionally, machine translation for a word or phrase that you are unsure about, to give you examples of how the word or phrase has been used elsewhere. For example, if you had to translate a industry-specific term like "photovoltaic adapter" or "hedge fund," you might search both translation memory and machine translation for matches.

  The standard and segment leverage translation memory search options are focused on finding exact and fuzzy matches, for the purpose of providing a translation for a complete segment. The concordance search is a secondary type search that attempts to find TM entries where the words provided have been used.

  See the "Concordance Search" topic for particulars.
- **Freeform SQL**. This mode, which is available from the **Advanced** link, enables you to author complex SQL-based queries against the translation memory. It supports the use of wildcard searches and complex groupings. In short, it supports all filtering clauses supported by the underlying database.

All search modes allow for search to use translation memory attributes to be used to filter the results. The freeform mode allows the greatest flexibility on how these can be used. Only the standard and segment leverage search modes can utilize word stemmers associated with the translation memory. (Please refer to the Word Stemming section for information on word stemming support.)

Search and Replace Tool

The search and replace tool provides a means for finding translation entries that may need to be corrected. Using the search and replace tool, you can find translation memory entries using a search mechanism that is similar to the standard search mode. The primary difference is that the search and replace tool does not support wildcard searches in the same manner as the standard search. Instead, you can disable the whole-word search option to provide some level of wildcard search.

Typical uses of the search and replace functionality include correcting common translation mistakes and making global labeling changes. The scope of the search and replace operation can be filtered using the same translation memory attribute filter options available in the standard and segment leverage search modes. Using these filter options, you can apply changes to the entire translation memory or a subset of entries based on a specific set of entries associated with a given import job (or multiple import jobs), a specific asset or a combination of other filter options.

After you identify a set of entries with search and replace candidates, you can apply the operation to all the records or to specific records that you select. Changes made to translation memory entries are applied only to the translation memory entries.

Note: Assets with segments that were leveraged from these entries or which were used to create these entries will not be updated simply by updating the translation memory. If the objective is to update content that has already been translated, you need to force the assets to be re-leveraged against the updated translation memory. It will not be sufficient to simply choose the Reapply TM option in the browser workbench or run any other similar SDK-based process because these processes will generally not update ICE or manually translated segments. To force a re-leverage of all segments, force the asset to be re-segmented, by clearing the segment cache (such as within a project workflow), changing the asset, or simply performing a process on the asset that modifies its timestamp.

Concordance Search Tool

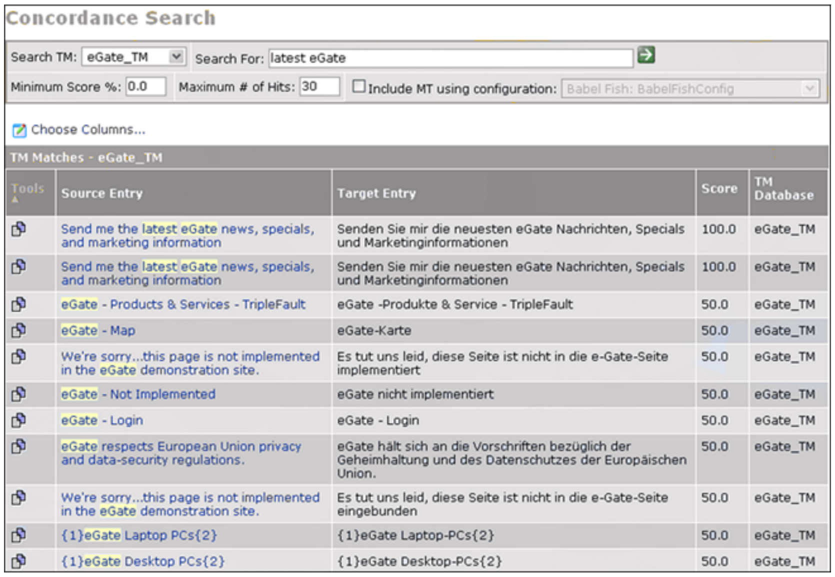
The concordance search tool lets you search translation memory and, optionally, machine translation for a word or phrase that you are unsure about, to give you examples of how the word or phrase has been used elsewhere. For example, if you had to translate an industry-specific term like "photovoltaic adapter" or "hedge fund," you might search both translation memory and machine translation for matches.

The concordance search tool can be run either as a Search Type from the WorldServer **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*** search page or from **Tools ➤ Concordance Search Tool** in the Browser Workbench. In the Browser Workbench, if you select a word or phrase, then **Tools ➤ Concordance Search Tool**, the **Concordance Search** dialog opens with all translation memory hits for that selected word or phrase. If you select a segment, by checking its check box, then invoke the **Concordance Search** dialog, it opens with that entire segment's contents in the search field, and returns all TM entries that have any of the words in that segment, sorted with those with the most matched words first. Typically, however, you would not search for an entire segment's contents, since that has been performed already when the asset was prepopulated from the TM.

Note: If you select a standalone TM in the **Search TM** selection box, the **TM Database** column also lists the name of the standalone TM for each match found. By contrast, if you select a TM Group in the **Search TM** selection box, the TM Database column lists the TM database name in which each match was found.

The standard and segment leverage translation memory search options are focused on finding exact and fuzzy matches, for the purpose of providing a translation for a complete segment. The concordance search is a secondary type search that attempts to find TM entries where the words provided have been used.

In the following search for the string `latest eGate`, WorldServer found two 100% matches—that is, matches of the entire string—and several 50% matches—where only one of the two words was found.



Search TM: eGate_TM Search For: latest eGate
Minimum Score %: 0.0 Maximum # of Hits: 30 ☐ Include MT using configuration: Babel Fish: BabelFishConfig

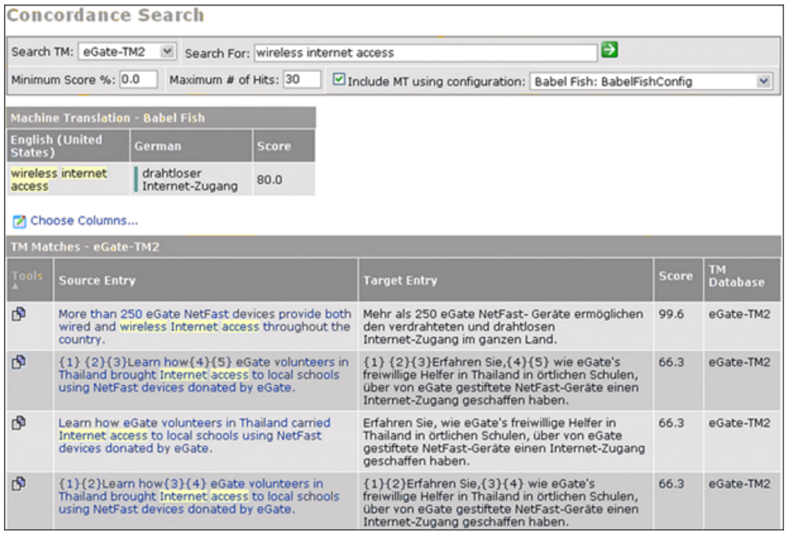
Tools	Source Entry	Target Entry	Score	TM Database
	Send me the latest eGate news, specials, and marketing information	Senden Sie mir die neuesten eGate Nachrichten, Specials und Marketinginformationen	100.0	eGate_TM
	Send me the latest eGate news, specials, and marketing information	Senden Sie mir die neuesten eGate Nachrichten, Specials und Marketinginformationen	100.0	eGate_TM
	eGate - Products & Services - TripleFault	eGate -Produkte & Service - TripleFault	50.0	eGate_TM
	eGate - Map	eGate-Karte	50.0	eGate_TM
	We're sorry...this page is not implemented in the eGate demonstration site.	Es tut uns leid, diese Seite ist nicht in die e-Gate-Seite implementiert	50.0	eGate_TM
	eGate - Not Implemented	eGate nicht implementiert	50.0	eGate_TM
	eGate - Login	eGate - Login	50.0	eGate_TM
	eGate respects European Union privacy and data-security regulations.	eGate hält sich an die Vorschriften bezüglich der Geheimhaltung und des Datenschutzes der Europäischen Union.	50.0	eGate_TM
	We're sorry...this page is not implemented in the eGate demonstration site.	Es tut uns leid, diese Seite ist nicht in die e-Gate-Seite eingebunden	50.0	eGate_TM
	{1}eGate Laptop PCs{2}	{1}eGate Laptop-PCs{2}	50.0	eGate_TM
	{1}eGate Desktop PCs{2}	{1}eGate Desktop-PCs{2}	50.0	eGate_TM

Figure 1: Concordance Search

You can select a translation memory and machine translation configuration to search in. The default TM and MT used is based on the configuration for the content being translated. Similarly, the source and target language is inferred from the asset in context.

The scores are displayed for each match using the "concordance scoring" method. The "concordance scoring" is the percentage of the sought words being found in the match, regardless of the number of other words in the segment. Normal fuzzy match scoring would have reported this as a much lower match because fuzzy match scoring is a percentage of the found words against the number of words in the segment.

The following example shows a concordance search when `Include MT using configuration <`*`MT configuration`*`>` is enabled.



Concordance Search

Search TM: eGate-TM2 Search For: wireless internet access

Minimum Score %: 0.0 Maximum # of Hits: 30 ✓ Include MT using configuration: Babel Fish: BabelFishConfig

Machine Translation - Babel Fish

English (United States)	German	Score
wireless internet access	drahtloser Internet-Zugang	80.0

Choose Columns...

TM Matches - eGate-TM2

Tools	Source Entry	Target Entry	Score	TM Database
	More than 250 eGate NetFast devices provide both wired and wireless Internet access throughout the country.	Mehr als 250 eGate NetFast- Geräte ermöglichen den verdrahteten und drahtlosen Internet-Zugang im ganzen Land.	99.6	eGate-TM2
	{1} {2}{3}Learn how{4}{5} eGate volunteers in Thailand brought Internet access to local schools using NetFast devices donated by eGate.	{1} {2}{3}Erfahren Sie,{4}{5} wie eGate's freiwillige Helfer in Thailand in örtlichen Schulen, über von eGate gestiftete NetFast-Geräte einen Internet-Zugang geschaffen haben.	66.3	eGate-TM2
	Learn how eGate volunteers in Thailand carried Internet access to local schools using NetFast devices donated by eGate.	Erfahren Sie, wie eGate's freiwillige Helfer in Thailand in örtlichen Schulen, über von eGate gestiftete NetFast-Geräte einen Internet-Zugang geschaffen haben.	66.3	eGate-TM2
	{1}{2}Learn how{3}{4} eGate volunteers in Thailand brought Internet access to local schools using NetFast devices donated by eGate.	{1}{2}Erfahren Sie,{3}{4} wie eGate's freiwillige Helfer in Thailand in örtlichen Schulen, über von eGate gestiftete NetFast-Geräte einen Internet-Zugang geschaffen haben.	66.3	eGate-TM2

Figure 2: Concordance Search Including Machine Translation

Machine translation matches return a concordance score equal to the level set for the `Fuzzy Score Equivalent` field when the MT adapter was configured. In this example, all concordance scores returned by the BabelFish configuration are `80.0`. This does not mean that you should use an MT match instead of any TM match with a score of less than `80.0`. The `Fuzzy Score Equivalent` is an advisory estimate of how your WorldServer administrator felt a BabelFish match would compare on the translation memory scale. However, since *every* match returned has the same score, this value should not be used as the basis for a decision.

Note: In future versions of the MT adapters, they will have guaranteed levels of quality (like High, Medium, Low), and WorldServer will be able to make programmatic decisions based on comparisons of the TM and MT scores.

The following rules apply to the concordance search:

- Concordance results are case insensitive. However, a penalty is assessed for capitalization penalties in the same way they are assessed for standard and segment leverage match results. See the WorldServer *Translation Memory Administration Guide* for details about the TM score capitalization penalty property.
- Concordance scoring uses only words and numbers. Placeholders and punctuations are ignored for scoring purposes.
- The `Minimum Score %` threshold is used to filter the results.
- The `Maximum # of Hits` value restricts the displayed result set. All hits are displayed in a single page, so you should limit results to a small set.
- Depending on your permissions, you are granted access to the TM entry editor by clicking on the source text of the result. This applies to TM based results only.
- Order and position of words in your search text is not significant. That is, searching for “cars and trucks” or “trucks and cars” both return a 100% match against the segment “His favorite modes of travel are cars, planes, and trucks.”
- Found search words are highlighted in the source text of the results. Target-based concordance searches are not supported.
- The sort order of the results is based first on the concordance score.
- Wild card searches are not supported.
- Searches are based on whole words or stems of those words if stemming is enabled. There is no support for substring or subword lookups.

Translation Entry Edit Tool

The WorldServer translation entry editor allows users to update existing translation entries. Everything from the source and target text to the custom attribute values can be updated. The user can also lock or unlock a given translation memory entry. The ability to access or edit a translation entry requires that the user have access privileges for the translation entry page.

Whenever changes are made to the translation entry, WorldServer creates a new history entry to provide an audit trail for all changes made to translation entries. The translation entry editor will provide access to the history entries, and provide a means through which history entries can be restored. This process simply requires that the history entry be opened, and that the you chose the Restore Translation option. This will create a new translation memory entry to replace to current active entry. The attributes from the historical entry will be restored, but will remain associated with the asset for which the current entry is associated.

The translation entry edit tool can be accessed from the entry links of both the search and search and replace results. Thus, you can fix unexpected issues that you discover while performing other operations.

Chapter 3

Understanding WorldServer Translation Memory Leveraging

Topics:

This chapter describes the basic use of translation memory for translations in WorldServer.

Leveraging and Segmentation

Leveraging is the process through which the translation memory (TM) is used to provide translations for content that needs to be translated. Segments from the asset are supplied to the TM engine. The segmentation process determines how content is broken up within the asset, and thus affects the leverage process. For optimal results, the asset filters need to employ the same segmentation strategy used in the creation of the TM. For instance, if the TM was formed from sentence-based segments, then the segmentation of content to be leveraged should also be segmented based on sentences. If the TM was created based on paragraph-level segments, then you would expect that attempting to leverage it against sentence level segments will result in low scoring matches.[2]

Impact of Segmentation Changes on Leverage

In an ideal world, the current strategies for segmenting assets would match the segmentation strategy used when the TM was first populated. Unfortunately, the TM world is not ideal, and the segmentation strategies are not always static. Here are some scenarios that can result in segmentation differences between the current asset segmentation strategies and those strategies used in the creation of the existing TM entries.

- Changes to filter segmentation
- Changes related to TM import
- Manual changes to segments during translation

Changing Filter Segmentation Rules

WorldServer allows customers to configure content filters to handle tags and other elements in a manner that seems appropriate for the customer. These options control how segments are formed during the segmentation process, and ultimately, how they are stored in the TM.

Changing the filter segmentation options may result in a noticeable change to leverage levels against the TM. New entries added to the TM after the filter changes will be consistent with the current filter strategies. However, the changes made to the filter may negatively affect the system’s ability to fully leverage the TM entries that were generated before the filter changes. Assets that were previously fully leveraged against the TM may no longer be fully leverageable against the TM. Because changes to the segmentation rules may affect the size, structure, and nature of the segments generated by the filter, these new segments will not necessarily match against the TM content generated from the old segmentation rules.

When it is possible, we encourage you to determine segmentation strategies in advance of any translation work, and not to change those strategies after work begins. This approach may not be realistic if new requirements or needs are discovered. Nonetheless, the impact of leverage based on these changes should be thoroughly considered. Before finalizing any segmentation changes or pushing them into a production environment, we encourage you to conduct tests to assess the potential impact. One way to run these tests is to export the production TM or TMs to a test environment, and run leverage projects before and after applying the proposed changes. (An ideal content set is one that is already 100% leverageable against the TM or TMs.)

[2] This ignores the impact of the auto-split/merge technologies discussed later because these technologies can be disabled and must be understood separately.

Migrating TMs from Different Sources

Often it is advantageous or necessary to migrate TM data from third-party sources or from other WorldServer sources. However, importing TM data into an existing TM or environment does not ensure that the TMs will be leveraged optimally. Segmentation rules are configured in each environment. If the environments in which the TMs were created are configured in the same way as the environment to which they are being migrated, then leverage should be optimal. Content that was previously fully leveraged against the imported TM should still tend toward 100% leveragability.

However, if there are differences in how the content was segmented during the creation of the TMs in the old environments, these differences will affect the WorldServer system's ability to completely leverage content that was formerly leveraged in its entirety against the TMs being imported. To fully leverage the TMs, the new environment will need to have its segmentation rules adjusted to match the TMs being imported. This task is often easier to suggest than to implement in that there are a number of factors to consider:

- How will the TMs be used? Should they be compiled into a single TM or separate TMs? Are they to provide backup data for translation or are they expected to provide the primary information for leverage?
- Is the migrated environment a new environment or an existing production environment?
- Do the segmentation rules used to create the TMs being imported represent the desired strategy?
- How important is it to be able to fully leverage old content? What is the expected leverage level against the migrated TMs?

The answers to the above questions will provide insight and guidance about how to configure and update the migrated environment. Toward this end, here are some important aspects of WorldServer to consider and remember:

- Segmentation rules are applied to filters, not TMs. TMs cannot tell the filter how to segment content for maximum leverage.
- WorldServer supports multiple filter configurations to be set up for each type of filter. These variations can be assigned to AIS paths (and inherited by sub-paths), and are applied to the filter(s) providing segmentation for those assets underneath the AIS paths. (The default configuration is applied to AIS paths that have not been overridden.)
- TMs are also associated with AIS paths. It is possible to provide an AIS path filter configuration that maximizes leverage against the TM that is assigned to the AIS path. Providing an optimal configuration for a TM group may be more difficult if the included TMs are not all based on the same segmentation strategy. If a TM group is being used, then generally the configuration should be biased toward the primary TM in the TM group.

Changing Segmentation Manually

If a qualifying TM match cannot be found for the original segment, the translator might consider splitting or merging segments to find existing matches that provide a more natural translation.

The following examples illustrate when you might want to split or merge the presented segments:

- A tag within the source of the asset has broken the segment at an unnatural language location. Merging this segment with the next one may facilitate the translation of a complete statement as opposed to a fragment.
- Two or more segments in the source language may be best translated as a single segment in the target language. (For instance, sometimes multiple sentences in one language may translate into a single sentence in a target language.)
- A portion of the content in a source segment may not need to be translated in the target (it might be irrelevant, or should be left in the source language). In this case it makes sense to split the source segment into smaller segments, isolating the text that should not be translated. The translator would then go on to translate the portions that should be translated, and clear the translation for the segment that should not be translated.

Splitting and merging segments changes the segmentation from what is configured by the filter segmentation rules. The new or modified segments will be presented to the TM for storage. The filters alone will not be able to replicate manual split or merged content in the future. However, through the automatic split/merge facilities described later, WorldServer can reproduce the manual segmentation introduced by users, provided that the appropriate options are enabled. (See *Auto Split/Merge* on page 71 for more information.)

Best Practices for Manual Segmentation

Manual splits and merges may provide better leverage against a TM when significant segmentation differences exist between the asset and the translation memory.

For example, you might consider using the split technology when the asset filter produces paragraph-level segments, but the translation memory mostly contains sentence-level entries. Breaking up the asset segments might make better matches against the TM entries.

Similarly, you might want to merge sentence-level asset segments when TM entries were created at the paragraph-level. The merge process can progressively merge successive asset segments to match against a larger TM entry.

Note: File type filters (which have FTS in their name) will not allow segment merging across paragraph boundaries.

Leveraging Across Different Types of Assets

As mentioned earlier, filters are aligned to specific types of assets. For instance, one filter will handle HTML or Markup type data. Another one will handle Microsoft Word documents. Text data will be handled by yet another filter. Each of these filters is developed to handle specific characteristics of the data it is to process. This data is often a combination of content and document-specific encodings or markup. The content is the actual text needing to be translated. The markup is used to define the structure and formatting of the content. Examples of markup are as follows: a hard or soft break, a piece of text rendered in bold, or even a marker for some object or invisible text.

Markup differs for data from different asset types. For example, a plain text file supports only a subset of the features supported by a rich text file (RTF). As this example illustrates, not all features are shared across different asset types. A user who is familiar with configuring filters knows that the filter needs to be told how to treat the various markup tags.[3] Some tags are treated as embeddable, while others are treated as non-embeddable (or breaking) tags. Embeddable tags do not lead to segment breaks, and are replaced in the segment by a placeholder.[4] Non-embeddable tags force segment breaks.

The presence and treatment of tags within asset data affect the segments generated. Handling of similar tags across different asset types affect the ability to leverage TM entries across document types. If you intend to work with asset types containing similar or identical content, take care to configure the filters for each asset type as closely as possible to an established segmentation strategy. This approach helps maximize the ability to leverage across asset type boundaries.

Leveraging Assets

Within WorldServer, content is presented to the user for translation. This content can be accessed from the WorldServer Browser Workbench or from offline translation benches such as WorldServer Desktop Workbench. Regardless of what tool you use to visualize and translate text, the underlying process is the same:

1. Identify content for translation.
2. If content is new or updated, run it through the segmentation and leverage process and store segmentation data.
3. Present stored segments and translations to the user for translation and review.

[3] Filter configuration and options are outside the scope of this document.
[4] The placeholder represents a marker within a segment for content that should not be translated, but rather, should be propagated to the translated document as-is.

Step 2 is the key step of interest at this point. WorldServer is able to determine whether content being requested is new or updated, based on the existence of segmentation data and change detection information for the asset. If the asset is new or updated, it must be filtered and leveraged. Based on the asset type, the appropriate filter is chosen to expose the segments. These segments are then passed into the TM to be leveraged. If a qualifying match is found, the translation is populated into the asset segment. If not, only an indication of the best matching score is stored with the segment data.

The leverage process evaluates a hierarchy of decreasingly valued matches. Matches are preferred in the following order:

1. ICE/SPICE matches
2. Exact matches
3. 100% Repaired Fuzzy Matches
4. Auto-translated/Machine Translation

If any of these matches occur, translations are automatically populated in the target segment of the asset. (Do not confuse this step with the pretranslate option, which populates the untranslated segments with the best fuzzy match in the TM.)

Reverse Leveraging

Translation memory reverse leverage is the ability to leverage TM content using the stored target information within a TM entry as if it were the source content, in order to provide translation support from the target language into the source language of the TM entry. In short, the TM reverse leverage operation reverses the source and target designations of segments within a TM entry, thus allowing for translation support of the target language into the source language.

When a TM is leveraged during the leverage process or during the pretranslation process, the system checks to see if the TM has been enabled to support reverse leveraging. If it has, the system first looks for TM entries that were stored for the desired language pair based on the source and target designations within the TM entries. If a qualifying match has not been found, then a reverse lookup is attempted. That is, forward-leverage matches are chosen over reverse-leverage matches from the same TM

If you are leveraging across a TM Group, reverse-leverage matches take precedence over equally scored matches from lesser ranked TMs in the group.

Reverse leveraging differs from "forward" leveraging (the default) in the following ways:

- It treats the target language of the match as the source language.
- It supports only potential 100% matches. SPICE and ICE matches are not returned as such.
- A configurable penalty is assessed to control auto-leverage.

  The `tm_score_reverse_leverage_penalty` TM property in `tm.properties` can be configured to penalize reverse leveraged matches. The default value is `0`. The allowed values must be in the range of `0` to `1`. This penalty is a leverage level penalty, and as such, it is applied to the entire match. (Many penalties are applied to the segment element—such as a word or number in the segment—that causes the penalty.) As a result, the impact of the penalty is not affected by the length of the segment. For example, if the penalty is set to `0.02`, then the final score will be reduced by `2` percentage points. Based on this value, the maximum score that a reverse leverage match could generate is a 98% score.

  The penalty would be applied to all matches resulting from the TM reverse leverage process. However, it will initially only apply to potential 100% and fuzzy matches since the TM reverse leverage process will not initially generate SPICE or ICE matches.

  You only need to add the `tm_score_reverse_leverage_penalty` TM property to the TM properties file when you want to override the default value. This property is currently only supported at the system level. Therefore, the assigned value will be applied to all reverse leverage matches across every TM.

Reverse leverage is useful if you have content in the same subject area being developed in multiple languages. For example, if you have some of your documentation written in English and some written in French, and the subject matter is similar, you could use common TMs for both. Without reverse leverage, only the matches in the direction you are translating to can be used. With reverse leverage, the matches in both directions can be used.

To illustrate this, consider the following example, in which the first TM (TM1) has four English to French segment pairs and one French to English pair, and a second TM (TM2) in this TM group has three French to English pairs (that is, TM2 is a bilingual TM, with entries in only one source language—French. TM1 has reverse leveraging enabled, allowing it to capture both English to French and French to English matches. And, finally, suppose TM1 is the primary TM in the group.

Table 1: TM Contents

TM Name	**Source Language**	**Target Language**	**Source Text**	**Target Text**
TM1	English	French	The cat in the hat.	Le chat dans le chapeau.
TM1	English	French	The 3 little bears.	Les 3 peu ours.
TM1	English	French	There is nothing like this anywhere.	Il n'y a rien comme ceci n'importe où.
TM1	English	French	Where are all of the children hiding?	D'où tous les enfants se cachent?
TM1	French	English	Les 3 peu ours.	3 little bears.
TM2	French	English	Pourquoi est-ce que tous les enfants se cachent?	Why are all of the children hiding?
TM2	French	English	Trop de cuisiniers dans le pot !	Too many cooks in the pot!
TM2	French	English	Le chat dans le chapeau.	A cat in the hat.

Suppose you translate content from French to English.

The following table compiles the lookup results for a number of search strings.

Table 2: Lookup Results

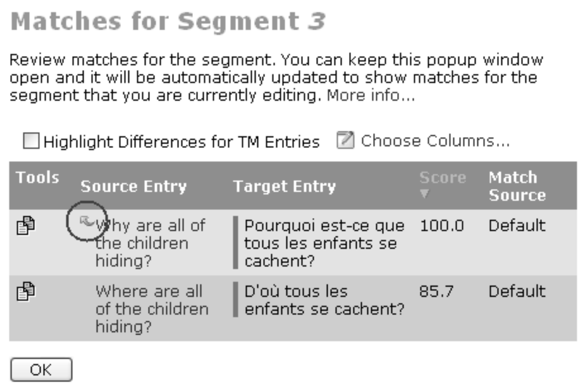
Lookup Text	**100% Matches** **(R) = Reverse**	**Preferred Match**	**Assigned As Reverse Match?**	**TM Used (and why))**
Le chat dans le chapeau.	(R) The cat in the hat. A cat in the hat.	The cat in the hat.	Yes.	TM1 provides a 100% reverse match. The reverse match was available from higher ranked TM, and so the forward match was not accepted.
Les 3 peu ours.	3 little bears. (R) The 3 little bears.	3 little bears.	No. Although a reverse match was available, there was a forward match available from the same TM.	TM 1. TM 1 also provides a 100% reverse match.
Il n'y a rien comme ceci n'importe où.	(R) There is nothing like this anywhere.	There is nothing like this anywhere.	Yes.	TM1.

Lookup Text	100% Matches (R) = Reverse	Preferred Match	Assigned As Reverse Match?	TM Used (and why))
D'où tous les enfants se cachent?	(R) Where are all of the children hiding? (Fuzzy match) Why are all of the children hiding?	Where are all of the children hiding?	Yes.	TM1.
Pourquoi est-ce que tous les enfants se cachent?	Why are all of the children hiding? (Fuzzy match) (R) Where are all of the children hiding?	Why are all of the children hiding?	No.	TM2.

This example applies the following leverage rules: Reverse leverage matches are ranked below other equally scored matches from the same TM. Forward or normal leveraged exact matches are ranked higher than reverse leveraged exact matches from the same TM. The same holds true for equally scored fuzzy matches.

Note the following about reverse leverage in the **WorldServer Browser Workbench.** :

The TM entries page from the Browser Workbench displays both normal and reverse leveraged matches. For example, if you perform a TM lookup in the Browser Workbench, you might see the following:



Matches for Segment 3

Review matches for the segment. You can keep this popup window open and it will be automatically updated to show matches for the segment that you are currently editing. More info...

Highlight Differences for TM Entries   Choose Columns...

Tools	Source Entry	Target Entry	Score	Match Source
	Why are all of the children hiding?	Pourquoi est-ce que tous les enfants se cachent?	100.0	Default
	Where are all of the children hiding?	D'où tous les enfants se cachent?	85.7	Default

OK

Figure 3: TM Lookup with Reverse Leverage

The reverse lookup icon () identifies the matches that result from a reverse leverage search. There is also a tooltip for the icon providing additional information.

WorldServer Rules for Ranking Matches

During the leverage process, there may be more than one appropriate match candidate. For instance, there may be multiple ICE matches or multiple exact matches. This section describes the rules with which TM matches will be ranked. These rankings apply for both exact and ICE matches. They do not, however, apply to SPICE matches, because there can never be more than one SPICE match in the system for a given segment. They also do not apply to fuzzy matches, because fuzzy matches are never automatically populated into segments during the leverage process.

The rankings are listed in order of precedence. The subsequent ranking criteria are leveraged only in the event that the higher-ranking criteria are unable to lead to a single choice.

1. **ICE Context:** ICE matches are ranked above non-ICE matches, and full context ICE matches are ranked above partial ICE matches. Exact matches are ranked below all ICE matches.
2. **Metadata-based Matching:** TM match entries that contain the same AIS-to-TM mapped attribute values as the asset being leveraged will be ranked above the other matches. Attribute matching is all-or-nothing; there is no support for matching just some of the attributes. All mapped TM attributes must match the values from the asset or else there is no elevation in ranking. For instance, if there are two mapped attributes, and a match from the TM contains attribute values that map to only one of the values from the asset, then this match is not treated as an attribute match.
3. **AIS Context:** Matches from the same asset are preferred over matches from other documents. If the asset in question is being re-translated, then the system prefers matches that were generated from a previous translation of the current asset.
4. **Position Rank:** Only used for ICE matches, and when the ICE match is from the same asset being leveraged. (Currently, this is used for re-translated assets only.) This ranking allows retranslation of a given asset to have multiple translations for a repeated segment, provided that the repeated segments were originally translated differently.
5. **Most Recent:** This is the final rank condition. Effectively, if all other things are equal, then take the most recent entry.

Chapter 4

Controlling Quality when Leveraging Translation Memories

Topics:

During the TM leverage process, WorldServer automatically populates the target segments of an asset if there is a qualifying TM match. A qualifying match is one that scores a 100% or registers as an ICE match. When a match is automatically populated into the target segment, it is considered to be *auto leveraged*. Auto leveraged translations are generally treated as high quality translations potentially needing to be reviewed. ICE matches are generally accepted as is, while 100% matches typically must be reviewed. This particular model hinges on two critical points: the expected quality of ICE matches and 100% matches, and the vendor arrangement. For example, your arrangement with the vendor might be that they review ICE matches, or an arrangement might be that they not review 100% matches.

This chapter describes two ways you can control the quality of your matches:

- Defining the context requirements for ICE matching
- Defining leverage level penalties

Defining Context for ICE Matching

SPICE matches and 100% matches are automatically applied to the segments. SPICE and ICE matches generally are not reviewed, while 100% matches are generally are reviewed. Fuzzy match segments must be translated entirely. The question that you must ask yourself is what makes a SPICE match or an ICE match worthy of not having to be reviewed. The implied answer is that the quality of ICE and SPICE matches are high enough that they do not require further reviewing. However, what is sufficient quality for one is not sufficient quality for another. If the collection of matches that fall into the ICE category includes matches that do in fact need to be reviewed, then all ICE matches would need to be reviewed in order to ensure the required quality of all translations. If it were possible to separate the matches that need to be reviewed from those that do not, then you would be able to save money by not having to review all of the matches.

In WorldServer 9, we have added a number of ICE leverage options. These options are designed to allow greater level controls over ICE matching criteria. The significance of this is that it enables the user to customize the requirements for ICE matches. The user can now enforce stricter requirement for ICE matches, enforcing higher level of quality to be associated with the ICE status. In WorldServer 9 the following options are available for further restricting ICE matches:

- **Asset match requirement** – ICE matches can be restricted to TM entries that are from a previous version of the asset currently being translated.
- **Metadata match requirement** – ICE matches can be restricted to TM entries that share a complete set of mapped attributes with the asset being translated.
- **Reviewed match requirement** – ICE matches can be restricted to TM entries that have been reviewed.
- **Full usage context requirement** – ICE matches can be restricted to TM entries that share a full usage context. This means that the previous and next segments associated with the TM entry must match those of the segment being translated.
- **Null (boundary) usage context allowance** – ICE matches can result for boundary segments based on leveraging an exact match that is on the same boundary. This null context or boundary condition is defined by the lack of a previous and/or next segment. The condition would allow a boundary segment to score an ICE based purely on the fact that the exact match being leveraged is on the same boundary as the segment (thus void of the same previous or next segment.) In short, this means that void or null is an acceptable usage context value.

Prior to WorldServer 9, ICE matching was based solely on the TM entry having been translated in the context of either the segment before or after the current segment being translated. The above option only influenced the ranking of ICE matches, but had no ability to promote an ICE match. These additional pieces of contextual data now have the ability to help define the context for an ICE match.

Asset Match Requirement (require_asset_match_for_ice)

This option allows you to impose the restriction that the ICE match process only considers TM entries that are associated with the asset (or an earlier version of the asset) being translated. Provided that an exclusive SID-based environment is not being used, WorldServer will store translations for each asset being translated. When re-leveraging or leveraging a newer version of the asset, the TM will prefer matches from the asset versus another asset based on the match ranking rules.

If the asset is new or has new segments, then those segments can still be ICE matched. The ICE matches in this case must originate from the translation of another asset. When dealing with a large volume of highly related information, this is extremely beneficial. It means that you can derive optimal benefit when leveraging newly introduced assets against an existing collection of TM data.

If your translation jobs center around the translation of unrelated assets or significantly different assets, then an ICE match generated from a translation in a different asset may not have the desired level of quality implied by an ICE match. This option allows you to extend the context requirement of ICE matches to include the asset.

This option does not affect SPICE matches. This asset match requirement is disabled by default.

Metadata Match Requirement (require_metadata_match_for_ice)

This option allows you to impose the restriction that the ICE match process only considers TM entries that share a complete set of mapped attributes with the asset being translated. Just as in the match ranking process, every mapped attribute must have the same value as those associated with the asset being translated. If even one of these differs, then the TM entry will not even consider for an ICE match.

The use of metadata allows you to describe and effectively partition your data within a single TM. Now, you can use metadata as an additional context requirement for ICE matching.

This does not apply to SPICE matches. This option is disabled by default.

When might you want to use this option to control ICE matching? Consider the following scenario. In WorldServer, you have the ability to consolidate your TMs into a centralized TM. What was once a collection of isolated TMs is now a single centralized, bilingual TM. The former collection of TMs may have included TMs for different products. Now that these TM are consolidated, the natural boundaries between translations from the products have been lost. If translations from different products are to be stored in the same TM, then how can we use of translations from one product being restricted in its use for another product?

As long as the TM entries satisfied the ICE matching criteria, it has the potential to score an ICE match regardless of which product it is associated with. The match ranking criteria will prefer matches from the same product over those from another product in the event that he ICE match exists for the desired product. This assumes that an AIS property for product has been mapped to a TM attribute for the product and that the ranking options for mapped attributes have been enabled. However, if there is no ICE match available from the desired product, then the ICE match available from another product will be selected. This standard behavior may not be desirable. Matches from a different product may require review regardless of the ICE match status. Because the collection of ICE matches can include ICE matches from a different product, then that all ICE matches must be reviewed in order to prevent ICE matches from a different product being accepted without review. If you are able to isolate the ICE matches that did not need to be reviewed from those that do, then you would be able to save money by not unnecessarily reviewing the ICE matches that do not require review.

In this example the problem can be solved by requiring the mapped metadata of the TM entry to match the mapped metadata of the asset being translated. By enabling this ICE matching requirement, matches from another product will no longer be considered for an ICE match. They will still be available as exact match candidates.

Reviewed Match Requirement (require_reviewed_status_for_ice)

This option allows you to impose the restriction that the ICE match process only considers TM entries that have the “Reviewed” status. This option is enabled by default, and is critical when using the Live TM functionality. When not in the Live TM mode, all entries stored to the TM are automatically stored as “Reviewed.” This option applies to SPICE and ICE matches.

While this option is exposed, SDL does not expect that you would choose to disable this. Nonetheless, you may have circumstances that may lead you to disable this option, if only temporarily.

Full usage context requirement (require_full_usage_context_for_ice)

This option allows you to impose the restriction that the ICE match process only considers TM entries that are a full usage context match to the segment being leverage. As noted earlier, ICE matching was based solely on the TM entry having been translated in the context of either the segment before or after the current segment being translated. The full usage context was only considered during the match ranking process, which means that in the absence of a full usage context match, a partial usage context ICE match could be leveraged for the ICE match.

This option simply provides you the means for further restricting the ICE match definition in order to enforce tougher context requirements. This option applies only to ICE matches, and is disabled by default.

Null (boundary) Usage Context Allowance (require_non_null_context_for_partial_ice)

ICE matches can result for boundary segments based on leveraging an exact match that is on the same boundary. This null context or boundary condition is defined by the lack of a previous and/or next segment. The condition would allow a boundary segment to score an ICE based purely on the fact that the exact match being leveraged is on the same boundary as the segment (thus void of the same previous or next segment.) In short, this means that void or null is an acceptable usage context value.

This option is disabled by default. When disabled, boundary ICE conditions are generally restricted to a partial usage context since only previous or next segments that exist are considered in the usage context definition. Enabling this option will allow the position of these segments to be significant, and result in possible full usage context matches.

Leverage Level Penalties

In addition to new ICE leverage options being added to WorldServer 9, new leverage level penalties have been added. Leverage level penalties affect the final match score based on factors that extend beyond source text differences. These penalties will allow you to restrict what will be considered an exact match and in most cases, these will affect ICE matches, and typically will not affect SPICE matches. Penalties that affect ICE matches will effectively cause such matches to be demoted not only from an ICE match, but the resulting match will result in a fuzzy match.

The following is a list of the supported leverage level penalties:

- **Maximum length penalty** – Penalizes matches where the length of the target text of the match exceeds the defined allowable limit for the target segment.
- **Unreviewed match penalty** – Penalizes matches that do not have the “Reviewed” status.
- **TM Group TM penalty** – Penalizes every match coming from a specific TM within a TM group, including SPICE and ICE matches.
- **Asset mismatch penalty** – Penalizes matches that are associated with a different asset.
- **Metadata mismatch penalty** – Penalizes matches that do not have a complete set of matching values with the asset for all AIS-TM mapped attributes.
- **Reverse leverage penalty** – Penalizes matches that are the result of the reverse leverage process.
- **Multiple exact match penalty –** Penalizes exact matches (excluding SPICE and ICE) where there are multiple distinct translations for the identical source text.

Each of the above penalties and how they are useful are described in greater detail later.

Maximum Length Penalty (maximum_target_length_penalty)

The maximum length penalty is applied to matches where the length of the target text of the match exceeds the defined allowable limit for the target segment. The maximum length scenario typically involves situations where content must be restricted to a confined area, such as on a display screen. Penalizing such matches can prevent the accidental use of a match that may have been truncated due to the maximum length restriction. Since providing a translation that actually meets the length restriction takes additional work, it makes sense that such matches should not be leveraged as 100%. The value of this penalty should reflect the effort on average of making the change.

Effective use of this penalty requires an environment that is able to define and apply maximum lengths for the target segment. WorldServer generally does not provide a solution out of the box for assigning or determining maximum lengths for targets. This is something that has to be driven by your business requirements and customizations. However, once you have integrated maximum length support into your WorldServer environment, the use of this penalty can help you control how TM matches treated during the leverage process when the target side exceeds the established maximum length criteria for a segment.

This penalty affects all matches, including SPICE and ICE.

Note: A property (`calculate_segment_length_in_bytes`) that you can configure in the `general.properties` file controls whether WorldServer measures segment length in bytes or characters. By default the length is measured in characters. Set this property to `true` to enable byte length calculation. The property is only applicable to maximum segment length limit functionality.

Unreviewed Match Penalty (tm_score_unreviewed_match_penalty)

This penalty is applied to a match that is being leveraged that does not have the "Reviewed" status. Depending on whether your WorldServer environment is using Live TM mode or not, you may have TM entries stored in your TM that do not have a reviewed status. Working in the non-Live TM mode, all matches stored to the TM are viewed. However, in Live TM mode, the stored TM entries may have various statuses, such as "Reviewed", "Unreviewed" or "Rejected" depending on what is happening within the project. (Refer to the documentation on Live Translation Memory.)

This penalty is a good way to control how the leverage treats matches that do not have the "Reviewed" status. By default, non-reviewed matches will not result in ICE matches, but they may lead to 100% matches. This is generally all right, since 100% matches typically must be reviewed anyway. During the leverage process, matches resulting from these TM entries will lead to the target segment being auto populated because it is a 100% match.

Perhaps your treatment of 100% matches is different and your goal does not require 100% matches to be reviewed. In order to do this, you need to ensure that the 100% matches that get registered meet some baseline quality guidelines. In this case, you might want to minimally require that only 100% matches that have been reviewed. This minimal goal can be accomplished by setting the Unreviewed penalty to a value greater than `0`.

Note: As you review the other leverage level penalties, there may be other opportunities for you to eliminate lesser quality 100% matches so that you end up with 100% matches that may not require further review, depending on your quality requirements.

This penalty affects all matches, including SPICE and ICE. If your intentions are to prevent ICE matches against unreviewed matches, then you should use the ICE restricting option instead.

TM Group TM Penalty

This penalty, which you configure in the user interface when you create or add to a TM Group, is applied to every match coming from a specific TM within a TM group, including SPICE and ICE matches. When configuring TMs within the TM group, you are able to assign penalties to each of the TMs. This penalty is not associated with the TM explicitly. Instead, they are related to the TM group's use of the TM.

The use of a TM group generally means that you have already chosen to use multiple TMs to partition your stored translation data. Perhaps you have chosen to create TMs for different products or different departments within your organization. Regardless of the strategy you use, there is a question you must answer whenever attempting to leverage TM content in a context that differs from the context in which the data was created. This question is, "How appropriate is this content for the context (such as a different product line) with which I want to cross leverage it?" If, in this example, the products are very related and share common terminology and language, then the use of the TM may be very appropriate. In such cases, you may choose to freely use it without requiring any special consideration. However , if the terminology and language are not very similar, or even worse, are contradictory, you may want to restrict the leverage being performed against. In this case, you can set a penalty that will be exacted against every match from that TM when being leveraged in that TM group.

In essence, TM group TM penalties provide a means to establish relative quality metrics across different TMs within the TM group.

Asset Mismatch Penalty (tm_score_asset_mismatch_penalty)

This penalty allows you to penalize a TM match if it is not associated with the asset that is being translated. When assets are translated, matches are created that associated with that asset. As a result, the next time the asset is translated, WorldServer is able to bias the selection of matches based on whether they are associated with the newer version of the asset. If this penalty is set to a value greater than zero, then the leverage process will also penalize matches that are not associated with the asset being translated.

Take care when using this penalty, as it affects all matches except SPICE matches. Nonetheless, this option may be useful to you if your quality requirements must take the source asset into consideration when evaluating ICE and exact matches. If your intentions are to prevent ICE matches against matches from a different asset, then you should use the ICE restricting option instead.

Metadata Mismatch Penalty (tm_score_metadata_mismatch_penalty)

This penalty allows you to penalize matches that do not have a complete set of matching values with the asset for all AIS-TM mapped attributes. Instead of using physically different TMs to partition translation data, metadata can be used. For instance, you can use a mapped attribute to track the product with which the TM entry is associated. If the products are significantly different in the terminology used or in the language, you might want to further restrict the extent to which translations are cross leveraged. The use of this penalty allows you to do that.

This penalty affects all matches except SPICE matches. If your intentions are to prevent ICE matches against matches where the mapped metadata differs, then you should use the ICE restricting option instead.

Reverse Leverage Penalty (tm_score_reverse_leverage_penalty)

This penalty allows you to penalize all reverse leverage matches. In WorldServer, TM content can be leveraged in both directions—source to target, and target to source. This is referred to as bi-directional leverage support. The maximum leverage level supported for reversed leverage matches is 100%. If a reverse leverage match results in a 100% match, it is treated the same way a normal, forward leveraged match—it will be populated into the target segment. Depending on your quality metrics, this may or may not be acceptable. You may want them to only be treated as high fuzzy matches. If this is the case, then this penalty can be used to prevent reverse leverage matches from generating 100% matches, and thus, prevent the target segment from being auto populated with them during the leverage process.

Multiple Exact Match Penalty (tm_score_multiple_exact_match_penalty)

This penalty allows you to penalize exact matches that occur when there are other exact matches for the segment with differing translations. Exact matches are ranked using the same rules for ICE matches, and generally will provide the best available match.

Nonetheless, perhaps the existence of multiple translations for a segment implies the introduction of a quality issue. There is already the option to detect multiple exact match segments; however, the target segment will still be populated with the system's best guess for the exact match. If you are not comfortable with WorldServer choosing the best exact match for you, you could choose to penalize such matches so that they would be available to the translator as high fuzzy matches (or whatever leverage level as dictated by the penalty) instead of them being positioned solely for review as an exact match.

The use of this option requires that the view multiple exact matches option is enabled. This penalty does not affect ICE matches.

Fuzzy Lookup: Quality *versus* Performance

The candidate selection process (which determines which TM matches will be returned and scored during the fuzzy match lookup) returns a number of candidates which is based on a configurable option. There are two such options. One controls the number of candidates considered during the leverage process. The other controls the number of candidates considered during the Browser Workbench and translation kit export lookup processes. By default, the two settings are not equal. The leverage option is set to return `5` matches, while the other option is set to return `20` matches.

The TM properties are:

```
# The maximum number of fuzzy results requested by the BWB and export code.
max_bwb_results=20
```

and

```
# The maximum number of fuzzy match candidates that are returned from database
max_fuzzy_results_in_asset_lookup=5
```

Setting the browser workbench setting to return more candidates than the leverage process can result in the user finding a better match for a segment when looking at matches in the translation workbench as compared to the scoping information. This can be useful for your translator who wants the best possible matches available. However, this can lead to somewhat confusing situations in which the best available match found during the leverage process is different from the best match available to the translator in the workbench. In the most extreme case, a segment may be leveraged at less than a 100% match, while the workbench results may show a 100% entry. While this can be slightly confusing, any discrepancy will be in the translator's favor; the translator will always see results that are equal to or better than the leveraged score.

You should *never* set `max_bwb_results` to be lower than `max_fuzzy_results_in_asset_lookup` as this can result in the translator not receiving the matches that correspond to the scoping information.

If you want to ensure consistency between the leverage process (for example, scoping results) and the matches users will see in the workbenches, these two values should be set to the same value. At the same time, raising the value of the `max_fuzzy_results_in_asset_lookup` will slow down the leverage process. Therefore, you should be very cautious when raising this setting.

Chapter

5

Controlling Translation Memory Security

Topics:

- *Translation Memory Security*

WorldServer provides three techniques that can be used in combination to impose a security policy for your translation memories:

- **User Type Permissions** – These permissions control what a user can see in the user interface.
- **Access Controls (ACLs)** – These permissions control whether a user or group has read or write access to specific translation memories.
- **Attribute Masks** – TM attribute masks control the visibility of TM entry and TM attributes.

This chapter describes how you can use these security measures to protect your translation memories.

Translation Memory Security

Translation memory in WorldServer has the following security features:

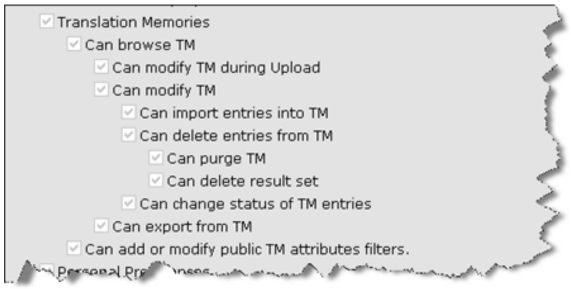
- You can restrict users' write permissions to updating or adding TM entries during the translation process, but disallow them to edit TM entries directly.
- You can set up users with limited modify permission: they can edit individual TM entries, but they can't execute wide-ranging, bulk operations like purging the TM database.
- You can define TM entry attributes that are specific to one or more TM databases. Someone editing a TM entry will see only attributes that are relevant to that TM database.
- You can limit access of a TM database to specific users or groups. Read and modify permissions can mix: someone might have modify permissions for one TM database but only read permission on another and absolutely no access to a third TM database. For example, you can allow a user to only browse or leverage against a TM, while that same user may have full modify permissions in another TM. You can assign permissions to a user group, thus allowing the users associated with that group to gain permissions or have their permissions restricted based on that user group. If the user is assigned explicitly to an object, then user groups can be used to add additional permissions, but not to restrict, since the users rights are not provided solely by the user group.

This capability is achieved through a combination of:

- User Type permissions
- Attribute masking
- Access Controls

User Type Permissions

The User Type permissions that control TM access are shown in this excerpt from the User Type permissions list for a user type in the **Management ➤ User Setup ➤ User Types ➤ User Type: <*Type*>** page. This excerpt is in the `Tools Tab` section.



- ✓ Translation Memories
  - ✓ Can browse TM
    - ✓ Can modify TM during Upload
    - ✓ Can modify TM
      - ✓ Can import entries into TM
      - ✓ Can delete entries from TM
        - ✓ Can purge TM
        - ✓ Can delete result set
      - ✓ Can change status of TM entries
    - ✓ Can export from TM
  - ✓ Can add or modify public TM attributes filters.

Figure 4: User Type Permissions Controlling Translation Memory Access

These permissions, in combination with ACL permissions, control the following:

- **Translation Memory** – Gives access to **Tools ➤ Translation Memories**
- **Can browse TM** – Allows you to browse a translation memory.
- **Can modify TM during upload** – allows you to modify translation memory entries only during the upload of a translation kit to WorldServer. If this option is set, when you upload the translation kit, you are presented with the additional option: **Update the WorldServer Translation Memory?** If you choose to do so, the kit import will update the TM. If you allow full access, users can modify, delete, and see other TM entries. If you switch all the other TM permissions off and only leave this one enabled, you limit users to uploading their own work for their own assets that they are assigned to work on.
- **Can modify TM** – Allows you to modify translation memory entries.
- **Can import entries into TM** – Controls the availability of the **Import TM from TMX File** button in **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*>**.

- **Can delete entries from TM** – controls the availability of the **Delete Selected** button in **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*>** that appears when you have searched and have a result set.
- **Can purge TM** – controls the availability of the **Purge TM Data** button in **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*>**.
- **Can delete result set** – controls the availability of the **Delete Result Set** button in **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*>** that appears when you have searched and have a result set. When you enable this button, the **Delete Selected** button is also enabled.
- **Can change status of TM entries** – TM entries have statuses when you are in Live TM mode. This permission governs a user's ability to change status for a TM entry.
- **Can export from TM** – controls the availability of the **Export TM as TMX File**, **Export Selected**, and **Export Result Set** buttons in **Tools ➤ Translation Memories ➤ Translation Memory: <*TM*>** that appear when you have searched and have a result set.
- **Can add or modify public TM attributes filters** – With this permission, you can choose to "Make this filter public" when you add or modify an attribute filter using the Attribute Filter Editor.

Access Controls for Translation Memory

In the **Access Control** tab, you can configure the visibility and modifiability of individual translation memories.

Note: ACLs do not apply to the out-of-the-box `Admin` user. The `Admin` user has access to all TMs, regardless of the ACLs defined.

By default, access control is disabled for translation memory (**Unrestricted**) meaning that only user type permissions apply. The permissions selector fields are not visible. If you select the **Restricted** option, the permissions selector fields appear. These permissions control what users or groups have read or write access to specific translation memories.

Note: These ACL permissions work in tandem with existing user type permissions. To browse a translation memory, your user type must have TM browse permission and read ACL permissions for that translation memory. Similarly, to be able to modify a TM entry, you must have TM modify permission for your user type and write ACL permissions for that translation memory.

Note: When a user saves a translation memory whose access mode is *Restricted*, WorldServer ensures that this user retains Read and Write Permissions. If the current user has been moved to the **Available Users** or **Read Only Permission** list, WorldServer inserts that user in the **Read and Write Permissions** list, and then saves the translation memory.

TM group permissions will be derived from the TMs used within the TM group. A user will be allowed read permissions to the TM group as long as one TM within the group allows read permissions to that user. In such cases, TM entries can only be read or leveraged from TMs that allow the user read permissions. Access to management of TM groups trumps ACL permissions. For instance, if the user requests a list of all TMs in the TM group, he is only given a sub-list of those TMs for which he has permission to read.

Similarly, a user can only write to a TM group if the user has write permissions to the TM configured as the write TM for that TM group.

Attribute Masking for Translation Memory

By default, all TM entry and TM attributes are visible in all translation memories. In a WorldServer installation with many independent translation memory instances, the cumulative number of attributes defined can be overwhelming. WorldServer allows you to control the visibility of TM entry and TM attributes in such situations so users of a translation memory will see only attributes relevant to that TM.

Under the **General** tab in the Translation Memory definition page, the `Attribute Mask` field controls the visibility of attributes. Radio buttons (**Restricted** and **Unrestricted**) indicate whether an attribute mask will be applied for that TM. By default, all TM entry attributes will be visible for that TM (**Unrestricted**). and the attribute selector fields not visible. If you select the **Restricted** option, the attribute selection fields appear, allowing you to select attributes that will be available when you import, add or edit TM entries.

When searching an individual TM with attribute masking, only those attributes relevant to that TM are available for searching. This affects the advanced search filter and Freeform SQL search. When searching a TM Group, a superset of attributes relevant to its individual TM is available for searching.

Summary Table of User Type and ACL Permissions for Translation Memory

The following table lists the user type permissions and TM access control permissions needed for each TM operation. User permissions are organized in a hierarchy: having a permission implies also having its parent permission.

Operation	**User Type Permission**	**ACL Permission**
Browse/search TM entries	Can browse TM	Read
Modify TM entries on upload	Can modify TM during upload	Write
Modifying TM entries	Can modify TM	Write
Importing TM entries	Can import entries into TM	Write
Deleting TM entries	Can delete entries from TM	Write
Purging entire TM	Can purge TM	Write
Deleting browse/search results	Can delete result set	Write
Manage public TM attribute filters	Can add or modify public TM attributes filters	Write
Export TM entries	Can export from TM	Read
Lock TM Entries	Can lock TM entries	Write
Unlock TM Entries	Can unlock TM entries	Write

Chapter 6

Sharing TM Memories in Live TM Mode

Topics:

One of the basic expectations of a modern centralized translation memory system is the ability for different translators to share translation memory in real- or near real-time. WorldServer 9.0 enables this capability while also ensuring that translators (and automated leveraging processes) can distinguish between entries that have and have not been reviewed.

This chapter describes a new approach to translation memory in WorldServer in which translation memories are “live.” Entries are added constantly during the translation process, and status is tracked to distinguish different types of entries.

Live Translation Memory Mode

In WorldServer 9.0, translation memories can be configured to operate in *live* mode. In live mode, entries are added to the translation memory constantly during the translation process, and status is tracked to distinguish different types of entries. Translation memories are updated whenever the segmented asset cache is updated. Any operation that updates the segmented asset cache will update the translation memory (for example, saves to AIS, uploads from desktop translation tools, and so on).

In addition to enabling real-time sharing of translation memory, live translation memory makes it possible to effectively update projects by restarting tasks without losing work.

Note: To enable live mode, go to `tm.properties`, uncomment the line:

```
#enable_live_translation_memory=false
```

and change `false` to `true`. After you restart WorldServer, you will be in live mode.

Live translation memory represents a substantial change from how translation memory worked in WorldServer 8.*n*. If you are an existing customer, you may have to make conceptual, process, workflow and permission changes to adopt live mode. WorldServer still defaults to non-live mode, where translation memory is updated approximately the same as in WorldServer 8.*n* versions, in case you prefer not to switch to live mode.

The *Translation Status* of a segment refers to if it is **Pending Review**, **Reviewed**, **Rejected**, or has no status.

Note: There is translation status on a *segment* (in the workbench and cache), and translation status on a *TM entry*. See the table *Table 3: Segment status in workbench and TM status* on page 52 for how these relate.

The combination of live translation memory mode and segment status makes it possible for translations to go into translation memory as soon as possible for the benefit of all translators, but at the same time translators can distinguish between translations that have been through a review process and those that might have just been added.

As translators work, the source and target text are copied from the segment cache to translation memory. Segment translation status determines if a segment is added to or updated in the translation memory and what translation status the corresponding entry should have. The following table shows what happens to the status in the translation memory when a segment's translation status changes in the workbench:

Table 3: Segment status in workbench and TM status

Segment translation status	Add or update made in TM?	TM entry translation status
None	No	No update
Pending Review	Yes	Pending Review
Reviewed	Yes	Reviewed
Rejected	Yes	Rejected

When WorldServer first segments your file and creates the bilingual segment list, it tries to fill in some of the translations from the translation memory. If it finds an ICE match or a 100% match it will automatically fill them in. (Note that by default only TM entries with a status of **Reviewed** can result in ICE matches.)

If WorldServer finds an ICE match, it fills in the target segment with it and automatically sets the segment translation status to **Reviewed**. If WorldServer finds a 100% match, it fills in the target segment with it and by default automatically sets the segment review status to **Pending Review**.

Note: You can reconfigure how you want WorldServer to treat 100% matches. You can change it to set the status to **None** or to **Reviewed** (not recommenced) by going to `tm.properties`, uncommenting the line:

```
#maximum_exact_translation_status=Pending
```

and changing `Pending` to `None` or `Reviewed`.

When a translator starts working, after she or he translates a segment and clicks **Save**, the translation goes into the translation memory as **Pending Review**. Later, if a reviewer changes the translation status of the segment in the segment cache (that is, in the workbench) to **Reviewed** and clicks **Save**, the corresponding entry in translation memory is also set to **Reviewed**.

By the end of the workflow, all segments should have a **Reviewed** status in the translation memory. When the reviewer sets the status of a segment to **Rejected**, the asset should go back to the translator. The translator can use the **All rejected** filter in the Browser Workbench to see only the rejected segments. When the translator fixes the rejected segment, he or she then changes the segment to **Pending Review** manually, using the **Tools ➤ Translation Status** menu. When the translator has fixed the rejected segments, the asset goes back to the reviewer, and this cycle continues until all segments are marked **Reviewed** by the reviewer.

Additional TM Properties for Live Translation Memory Mode

In addition to the `enable_live_translation_memory` and `maximum_exact_translation_status` properties already described, there are two other settings in `tm.properties` with which you can control live translation memory mode behavior. These properties are commented out by default.

- `require_reviewed_status_for_ice=true` – Determines if translation entries must have **Reviewed** translation status as a requirement for ICE matches. The default behavior (which is, effectively, `true`) is that unreviewed translation memory entries do not satisfy the ICE criteria. To make **Pending Review** TM entries with ICE match scores have their segment translation status set to **Pending Review** (instead of **Reviewed**), uncomment this property and set it to `false`. In non-live mode, all translation memory entries are reviewed, so ICE matches are unaffected by this configuration.
- `tm_score_unreviewed_match_penalty=0` – Defines the penalty for TM matches with an unreviewed translation status. In non-live mode, all translation memory entries are reviewed, so ICE matches are unaffected by this configuration. This is a leverage level penalty. It is applied to the final score produced by the match, and is only applied during leverage based processes. The value should be between 0 and 1. This penalty applies to all TM matches.

Migrating to Live Translation Memory Mode

When you upgrade an existing system to WorldServer 9.0 and enable live translation memory mode, you must be aware of a number of considerations.

When you upgrade to WorldServer 9.0, the status of all of your TM entries is set to **Reviewed**. Also, in non-live mode, whenever an entry is written to the translation memory it is given a reviewed status. This holds for all places that the translation memory is updated, including the Browser Workbench, import, and automatic actions such as `Save`. This helps you learn that entries added to translation memory the old way are considered “Reviewed” and also ensures that leverage works correctly when you first enable live mode.

Note: There is no need to specifically change leverage behavior, scoring, or concordance search for non-live mode, since every entry in the translation memory has a status of **Reviewed**. Any penalty or special behavior based on TM entries with non-Reviewed statuses therefore never occurs.

Because you might have workflow customizations that make explicit calls to update the TM from a segment cache, WorldServer checks for such explicit calls when live mode is enabled. If it detects a call to update the translation memory, it logs a warning indicating that the approach is deprecated. This is intended to help you find parts of your configuration that are not taking advantage of or are incompatible with live mode. As an example, you might have a `Save` automatic action with `Update TM` enabled.

Consider the following:

- In the old model, your translation memory was only updated when you specifically indicated to update the TM or if you put certain automatic actions such as "save" in your workflow. Your main concern was making sure that everything got into TM by the end of your projects. In the new model any segment with a translation status other than `none` will go into the translation memory. Your main concern is making sure that by the end of your workflow all segments are set to **Reviewed**. Look at the instructions you give your users and your workflows with this in mind, to figure out what you need to change.
- If you are currently using `Save` with `Update TM` at the end of your workflows to ensure that the TM is updated, consider adding a `Set Translation Status` auto action step just prior to `Save`, to set all segment translation statuses to **Reviewed**.
- If you are using TRADOS or some other tool, on import all translated segments will be set to **Pending Review**. If you consider the work your LSP does to be “Reviewed”, then put a `Set Translation Status` auto action in the workflow after the LSP to set all segments to **Reviewed**.
- Do not be concerned if you realize you have made a mistake and a project finishes but some or all of the segments were never set to **Reviewed**. Use the translation memory management page to find the appropriate entries in TM and set them to **Reviewed**. Unlike the old model, you rarely need to worry about completely losing translation memory.
- Monitor your log files for warnings. If you are using steps in your workflow that are explicitly updating translation memory, you will receive a warning. You can use these warnings to improve your process.

Differences Between Live Translation Memory Mode and Non-Live Mode

The following table summarizes some of the behavior differences between live translation memory mode and non-live mode.

Area	Live Mode On	Live Mode Off
TM Update	Whenever segment cache is touched.	On explicit calls to update TM. (Note that all entries added to TM are set to **Reviewed**.)
Browser Workbench	No **Save and Update TM** button.	Has a **Save and Update TM** button as in previous WorldServer versions.
Desktop Workbench: sending of segments in real-time to server	Allowed (Desktop Workbench can send segments in real-time to get updated in segment cache and translation memory).	Not allowed.
XLZ Import	No **Update the translation memory using imported assets** check box.	Has an **Update the translation memory using imported assets** check box as in previous versions of WorldServer.
Import of other translation kit formats (besides XLZ)	Segments with a `none` status are set to **Pending Review**.	No special behavior.
User type permissions and ACL privileges	The following user type permissions are meaningless and hidden: • Can save asset and update TM • Can modify TM during Upload If a user that has only Read ACL privilege for a TM opens an asset in the Browser Workbench, the	Same as in previous versions of WorldServer.

Area	**Live Mode On**	**Live Mode Off**
	workbench does not present a **Save** button, and issues this message: `WARNING: You do not have permission to update translation memory "<TM>". Write permission is required when using Live Translation Memory. Please, contact your WorldServer Administrator.` For the user to be able to work in the Browser Workbench in Live TM mode, the administrator must give that user Write ACL privilege for that TM.	
Logging	Anytime `WSAssetTranslation.saveTranslationToTm()` is called, a warning is logged with a stack trace.	No special behavior.
Desktop Workbench	No **Update the WorldServer Translation Memory** check box, nor a warning on import that the setting is being ignored.	Has check box.
Leverage	TM entry translation status affects segment status.	TM entry translation status does not affect segment status unless **Segment Asset** automatic action is used with `Set Translation Statuses?` and `Maximum Translation Status for 100% Matches` arguments set to `Yes` and `Pending`.

Chapter

7

Translation Memory Migration

Topics:

Many WorldServer prospects and customers have legacy TM data originating from other third-party TM systems. Often, this data is the result of years of translation efforts. While WorldServer provides greater control over their translation processes, it would not be economically feasible for customers to lose the ability to effectively leverage their historical TM data. In the past, moving TM data from one TM system to another could result in a substantial loss in leverage, and thus, additional translation costs.

SDL has invested substantial time and development into improving the migration process involving TMs from products such as TRADOS, Star Transit, and SDLX. Now customers can enjoy the benefits of the WorldServer product with minimal loss in leverage of their historical TMs. In fact, some customers have benefited from increased leverage rates due to related WorldServer TM technologies, such as the automatic split/merge and segment repair technologies (described elsewhere in this document).

The WorldServer product provides two tools for assisting in the migration of legacy TM and translation data: the asset alignment automatic action, and the TMX migration utility. The choice of which tool to use depends on the format in which the translated content is available.

- Use asset alignment if the source and target assets are all that is available.
- Migrate TMX files, with the stored TM data, from third-party tools in two ways. Import files directly if they are in 1.4b format. If they are not, use the `Trados2Idiom` utility to process the files before import.

An automatic WorldServer process performs alignment validation on the source and target segments when SID values are available. This ability can be modified as described in *SID-Based Alignment* on page 63.

Asset Alignment

Asset alignment is the process through which a source asset is aligned segment-by-segment to a translated asset. The product of the alignment is a collection of translation pairs that can be stored in the TM. This process can support the generation of TM data from a collection of translated assets. Conceptually, it makes sense that if you have both a translated document and the original source document, then it should be possible to match them up. In practice, this process is extremely complicated and yields varying results.

WorldServer provides an asset alignment tool that assists in the creation of new TM entries through the alignment of the original source asset and the translated asset. The ability of this alignment process is quite limited, and the effectiveness is highly dependent on the nature of the data being aligned. For instance, the following characteristics will affect the alignment process:

- The lack of markup
- Rearrangement of content in the target asset
- Substitution of tags in the target asset
- Omission or addition of tags in the target
- Segmentation configuration of the utilized filters

These are just a few factors that affect the ability to align assets automatically. The implemented heuristics go a long way to handle certain ideal, well-bounded cases, but overall the success rates are still generally low.

When to Use Asset Alignment

Depending on how translations were managed prior to the use of WorldServer, translation data may not have been stored or exported in a transferable format. While some systems leverage or support a common exchange format, such as TMX, other products may store data in a proprietary manner. If this is the case, the only accessible form of the translated content may be the actual translated documents. This process is valuable as a last resort. While it is not as effective as working from a TMX file, it can greatly decrease the translation costs when compared to having to start from ground zero.

Note that the alignment process is most effective with structured content with boundaries defined consistently by common markup. If the markup matches between the source and target, and the segments are not rearranged, the results may be more promising. Conversely, the alignment may not be reliable for plain text data, particularly if segments are rearranged on the target side and if the sentences do not map one-to-one between the source and target assets.

The Asset Alignment Process

The alignment process is accomplished in these steps: automatic alignment, manual align/review, and TM update. These steps are managed through a WorldServer translation project. The critical components of this project are the `Align` automatic action followed by a human translation and review step, and finally a TM update step.

The `Align` automatic action processes the segments generated for both the source and target assets. It then attempts to align them based on the implemented heuristics. It identifies the segments that are deemed to be aligned effectively, and marks them as being manually translated. For other segments, it will make a best guess or leave the target side completely blank (or simply populate it with appropriate placeholders.) Those segments for which it makes guesses are not marked as translated in the target asset.

To make the abstraction a little clearer, these assets are in play in the alignment process:

- The source asset containing the content to be translated.
- The target asset, which will be generated during the alignment process.
- The translated target file from which the alignment process attempts to match up to the source asset on a segment-by-segment basis.

After the `Align` automatic action completes, the next step is the translation step or the review step. Depending on the nature of the project, the translator may review the segments that have been aligned for correctness and then provide translations for the remaining segments. Upon completion, the translator may save the changes only to the target asset or may choose to update the TM as well if there is not a separate reviewer step defined in the workflow.

How to Use the Alignment Automatic Action

The basic workflow involving the `Align` automatic action is illustrated below.

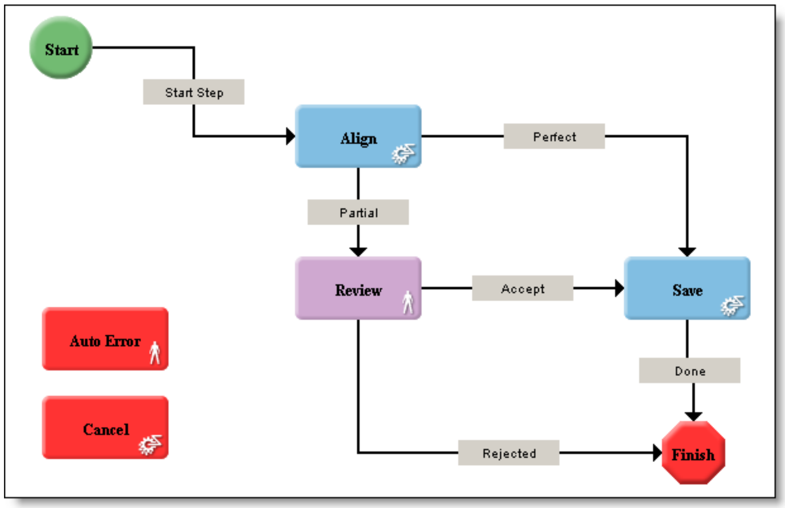


Figure 5: Alignment Automatic Action

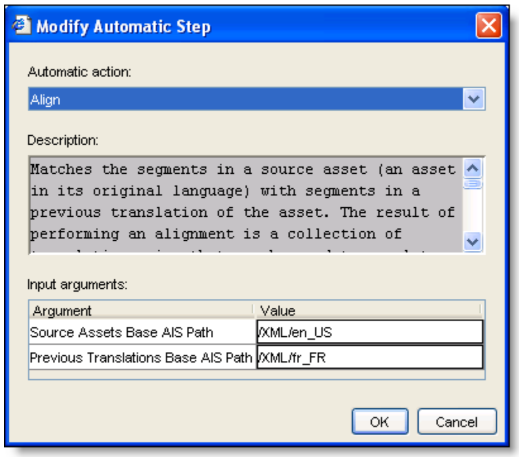
In short, the workflow has three primary steps – the `Align` automatic action step, the `Review` human step, and a `Save` automatic action step.

The Align Step

The `Align` action has two outputs or transitions defined.

- *Partial* – denotes that the asset was not completely aligned, and that some segments need to be translated and reviewed.
- *Perfect* – denotes that all of the segments in the source document were successfully aligned with the segments from the translated target asset.

Depending on the result, the asset will be reviewed or sent to the Save step. The `Align` automatic action requires two parameters to be configured.



Automatic action: Align

Description:

```
Matches the segments in a source asset (an asset
in its original language) with segments in a
previous translation of the asset. The result of
performing an alignment is a collection of
```

Input arguments:

Argument	Value
Source Assets Base AIS Path	/XML/en_US
Previous Translations Base AIS Path	/XML/fr_FR

Figure 6: Modify Automatic Step

These parameters define the base locations for the source assets and the translated assets to be used during the alignment process.

- *Source Asset Base AIS Path* – specifies the AIS base path for the assets to be aligned. This workflow can be used only with assets for which the base path is the first part of their AIS path. In the example above, the base path is set to `/XML/en_US`.
- *Previous Translations Base AIS Path* – specifies the AIS base path for locating the translated version of the source assets. In the example above, this is set to `/XML/fr_FR`. The translated files must match the name and relative path of the source assets. For instance, if the source asset to be aligned is located at the AIS path of `/XML/en_US/release1/index.html`, then the system would expect the corresponding translated target asset to be located at `/XML/fr_FR/release1/index.html`. (This example assumes that the base directories are defined as above.)

  Note: If this path is the same as the linked WorldServer target path for the given source path, then the original translated asset will be overwritten by the aligned version when the results of the alignment process are saved.

  Tip: The Align automatic action suppresses TM leverage during segmentation. If you create a workflow using the Align automatic action, we recommend that you not use the Segment Asset automatic action. Using the Segment Asset action causes TM leverage to occur against the source file before the alignment process runs. Enabling TM leverage during alignment reduces the effectiveness of asset alignment because TM content can affect the segmentation of the source asset. Also, the Align action may report partial alignment of the files, even if the unaligned segments were leveraged by the TM or the auto-translation process.

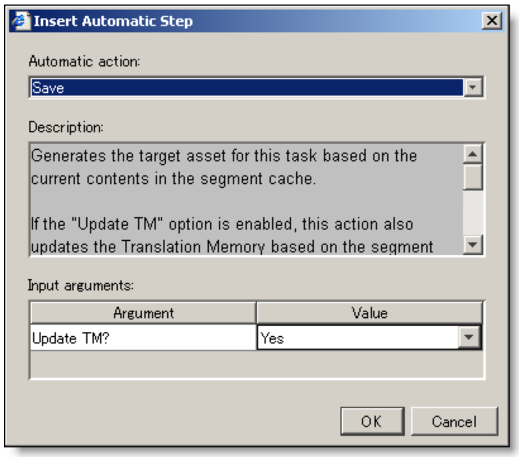
The Review Step

The review step allows for a user to check the results of the alignment process. In the workflow defined above, the review happens only when there is a partial alignment. However, typically, the review would happen regardless. The alignment process is not flawless, even when it reports a perfect alignment. Additionally, this step also provides the opportunity to perform additional translations for the unaligned segments.

Another variation to the above workflow is to define a translation step to follow partially aligned assets. The translation step and the perfect align transition would both then flow into a review step. If the reviewer rejects the work, it could be sent back to the translator instead of simply finishing.

The Save Step

The basic idea is to save the results, and if desired, update the TM with the newly aligned and reviewed translations. Saving the results merely causes the actual target file to be generated. This is not the same file that is used during the alignment process. It is the normal AIS target file based on the defined AIS target linkage definitions. Optionally, this step can update the TM with the provided translations. To do this, set the `Update TM?` parameter of the `Save` automatic action to `Yes`.



Automatic action: Save

Description: Generates the target asset for this task based on the current contents in the segment cache.

If the "Update TM" option is enabled, this action also updates the Translation Memory based on the segment

Input arguments:

Argument	Value
Update TM?	Yes

Figure 7: Update TM on Save

TMX Migration

Often legacy TM data can be exported from third-party tools into a format that is supported by WorldServer. Currently, WorldServer only supports the importing of TM data from TMX files. Provided the data is available in this format, WorldServer can import the data and make it available for leveraging within the WorldServer environment.

TM migration is most effective when TMX files are the source of the migrated data. However, simply having the data available in a TMX format does not ensure the highest level of leveragability once the data is imported into the WorldServer environment. While the LISA TMX specifications define the structure of the TMX files, and the expected usage of the supported XML tags, several issues affect WorldServer's ability to optimally leverage the imported data. Among these issues are the following:

- *WorldServer TMX compliance* – WorldServer is currently TMX Level 1 compliant
- *Third-Party Compliance Level* – TRADOS, while providing TMX export capabilities, is not TMX certified

Note: For the best results, export files into the TMX 1.4b format from all third-party tools. You can then import the files directly into WorldServer. Experience will tell you whether you might get some additional benefit from using the `Trados2Idiom` migration utility on TMX 1.4b files, but generally it is not necessary.

Directly Importing Third-Party TMX Files

Good results can be expected when you use the standard online import process for third-party files in TMX 1.4b format. However, be aware that the online process does not support the following options:

- Suppression of non-guaranteed TM aligned entries
- Ability to capture statistics on the processed entries

The impact of these limitations may not be significant enough to warrant use of command-line utilities as a normal process. You can always use the `Trados2Idiom` utility to get a profile of your data. The results can help you determine which path to take when migrating third-party TMs.

Note: Data exported from WorldServer never needs processing with the `Trados2Idiom` utility.

Improved TMX Import Quality

SDL has worked to improve the quality of the TMX entry alignment when importing third party TMX files. In some cases TMX entries imported from third party TMX files are not always aligned in a guaranteed manner. As a result, there can be a risk of importing and leveraging TM entries that should not be considered perfect matches. Typical scenarios include the following:

- **Source and target do not have the same number of embedded tags or placeholders**. The import process attempts to repair the entry by adding placeholders to the end of the segment with the missing placeholders. (This is required since WorldServer requires the source and target to contain the same number of placeholders.) While this makes the entry importable, it does not guarantee that the added placeholders are at the proper place. As result, this entry should either be suppressed or prevented from yielding a 100% match since the placeholder repair process is not currently capable identify the current place to insert added placeholders.
- **The source and target contain multiple substituted tags**. TM technologies like TRADOS allow the source and target to contain different markup. However, TRADOS-generated TMX files do not provide TMX information that would allow the tags in the source to be definitively aligned to the substituted tags in the target. In the event that there is only one substituted singleton and/or one substituted paired tag, WorldServer can determine the proper alignment. However, when there are more than one of the singleton tag or more than one paired tag substituted, the alignment is no longer guaranteed.
- **The source and/or target contains duplicated tags.** If the tags are identical, there is no issue since the targets will be generated identically regardless of which instance is ordered first. However, the impact of this goes beyond the alignment process in that the source could change in the asset leading to uncertain results. The problem is that there

is no clear way to know which repeated tags in the source should be mapped to the same repeated tag in the target. Similarly, if one of the repeated tags was substituted in the target for a different tag, it is not possible to know which one should be mapped to the substituted tag.

The alignment of tags should not be considered guaranteed unless the mapping of the tags in the source can be mapped to the tags in the target in a definitive way. In the past, the above entries were allowed to reach the TM, thus having the potential to lower the overall quality of the TM. In WorldServer 7.5.1 Service Pack 2, SDL introduced the distinction of non-guaranteed TMX entries, and implemented an enhancement that prevents any TMX entry deemed non-guaranteed from generating a 100% match. They can in fact lead to high fuzzy matches. When TMX files are imported in current versions of WorldServer, the alignment process prepends a series of special characters to the source segment text. As a note to the translator, the alignment process prepends a string to the target segment text as well, which notifies the translator that the placeholder order should be checked.

The duplicated tag scenario is a fairly common scenario, and the duplication of certain tags does not create any risk. However, you should decide which tags are to be considered unsafe and treated as non-guaranteed for this scenario. By default, all duplicated tags will result in a non-guaranteed TMX entry unless specifically identified as being safe. In order to ignore certain tags, these tags will need to be listed in the optional `ignoredTags.properties` file.

The `ignoredTags.properties` file must be created and then saved in the `ws\WEB-INF\classes\config` directory. Each tag name should be listed on a separate line, for example:

```
b

th

td

strong

br

em
```

You will need to create this file. Note that the above examples represent tags that SDL feels should be among the tags ignored.

Using the TMX Migration Utility

WorldServer provides a TMX migration utility in the form of a Java class named `Trados2Idiom`. Although named for the TRADOS product, it is used for migrating from other third-party TMX files as well. If the files are in a TMX format preceding 1.4b, they must be adjusted slightly so they can be properly imported into WorldServer. Often, this requires preprocessing the TMX files to address data aspects that are tool dependent.

To run the `Trados2Idiom` utility, run the following command on the WorldServer host:

```
java com.idiominc.ws.autoalignment.Trados2Idiom
 <-guaranteedEntriesOnly> <-showStatsSegments>
 <-statsFile file> <TRADOS
TMX file> <error file>
 <WorldServer-friendly TMX file>
```

Option Descriptions:

- *guaranteedEntriesOnly* – (optional) If this flag is provided, entries that are not considered guaranteed will not be exported to the newly generated WorldServer TMX file. *Not providing this flag will lead to all valid TM entries being exported to the new WorldServer-friendly TMX*. Not all entries marked as being non-guaranteed are necessarily aligned incorrectly. Rather, the heuristics employed simply cannot guarantee the correctness. Suppressing non-guaranteed entries may mean filtering out what may amount to good TM entries. However, it also means keeping out what may be bad entries. SDL encourages customers to apply this flag. While it will affect the leverage of old data, it is the safest approach. Nonetheless, customers are also encouraged to assess the impact of this flag, and make their own informed decision.

- *showStatsSegments* – (optional) This option tells the statistics engine to capture segment data for each of the counters so that they can be exported to the stats file. This allows you to identify segments that fit into certain scenarios (such as which had tags swapped, which are marked as guaranteed, etc.)
- *statsFile* – (optional) Path and file name of the file used to collect statistics data.

  The statistics include the number of entries exported, the count of swap and substituted tag entries, count of entries with placeholder repairs, and the number of non-guaranteed entries.

  For example, here is a sample from a status file with the showStatsSegments option:

```
Total entries with swapped tags only: 6
Total entries with tag substitutions (with possible tag swaps): 6
Total non-guaranteed entries : 6
Total PH range issue entries : 0
Total PH repaired entries : 1
Total invalid or empty TMX entries : 0
Total exported entries : 6 of 6
```

Note: Do not use this process on TMX files originating from WorldServer. It is unnecessary, and will generally result in an empty TMX file.

Definitions

- **swapped tags:** entries having tags that have been shifted around during the translation
- **substituted tags:** entries having tags that have been exchanged for another tag
- **non-guaranteed:** entries having tags that cannot be definitively aligned by the current implementation. This includes segments requiring placeholder repairs, segments with more than one singleton or paired tag substitutions, or segments with tags having different attribute values. Use the `guaranteedEntriesOnly` option described above to suppress these from being exported.
- **PH repaired:** entries that required placeholder repairing between the source and target segments.
- **PH range issue:** this is an external tracking stat for engineering only.
- **Invalid or empty :** entries in the TMX that are either empty or are missing a segment (source or target.) Wordless entries are considered empty.
- **Exported:** total number of entries exported. This is generally the total number of TMX entries minus the invalid entries count, and optionally minus the non-guaranteed if the `guaranteedEntriesOnly` option is provided.

Example

```
java com.idiominc.ws.autoalignment.Trados2Idiom
-guaranteedEntriesOnly –showStatsSegments -statsFile
stats_amv.tmx.log amv.tmx err_amv.log > ws_amv.tmx
```

The error file contains information about segments that the utility was not 100% about. Currently this includes:

- **alignment issues:** the number of tags do not match between the source and target segments.
- **tag mismatches:** tags differ either by name or attribute value.

SID-Based Alignment

The WorldServer alignment process will perform alignment validation on the source and target segments when SID values are available. The condition is simple:

- If the source and target both have SID values and they are not equal, do not align.

The presence of matching SID values does not guarantee a perfect alignment. Placeholder alignment is still required within the segment. If either of the segments is missing a SID value, then the SID condition is ignored. The segments may become aligned on account of the additional alignment engine algorithms.

In the event that this functionality needs to be disabled, you can modify the following TM property:

- **Description:** Determines whether SIDs are used during the alignment process if they are available. SIDs must match in order for the segments to be aligned if both segments contain SID values.
- **Name:** `enable_sid_based_alignment=true` (default)

Chapter

8

Translation Memory (TM) Groups

Topics:

Translation memory groups are a means of associating multiple translation memories under a single organizational unit. A translation memory group offers the ability to look up a text in multiple translation memories at once.

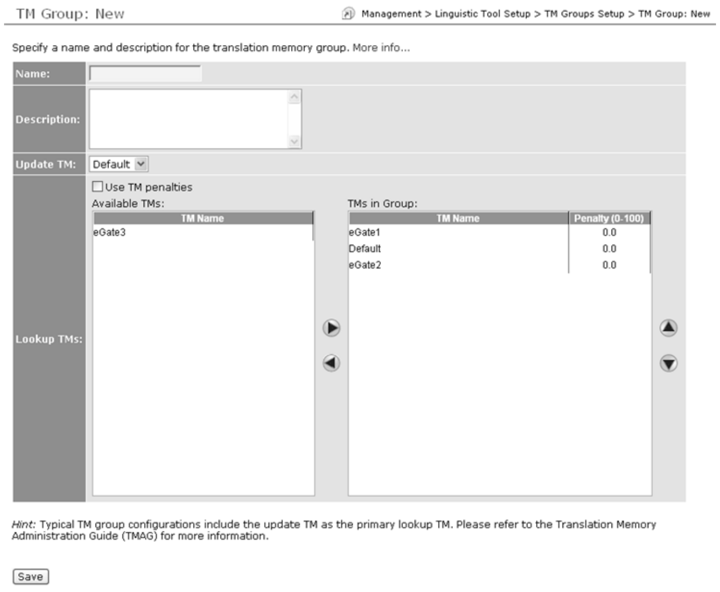
This chapter describes how to use TM groups in your WorldServer system.

Translation Groups Overview

In WorldServer, TMs are generally associated with AIS paths. This association identifies the TM to use for leveraging content stored at the AIS path location. It also identifies the TM to be updated when new translations are supplied for this same content. Only one TM selection can be associated to an AIS path, which limits the ability to leverage translation information stored in another TM. The customer has the option of storing all translation information into a single TM, but for business reasons, this may not be appropriate. Nonetheless, the user may need to leverage across multiple TMs. The TM group feature addresses this consideration and allows users to group TMs together so that they can be leveraged as a single entity. As a result, instead of associating a TM with an AIS path, the user can specify a TM group.

The WorldServer TM group implementation is based on rank. Within the TM group definition, TMs are not given equal weight. When you create a TM group, you need to choose the TMs to be included, and also establish the order of precedence. Additionally, only one TM in the TM group can be designated as the TM to receive new or updated translations. This TM is called the *update TM* or *write TM*.

The following figure shows the user interface for creating a TM Group within WorldServer.



TM Group: New

Management > Linguistic Tool Setup > TM Groups Setup > TM Group: New

Specify a name and description for the translation memory group. More info...

Name:	
Description:	
Update TM:	Default
Lookup TMs:	☐ Use TM penalties

Available TMs:

TM Name
eGate3

TMs in Group:

TM Name	Penalty (0-100)
eGate1	0.0
Default	0.0
eGate2	0.0

Hint: Typical TM group configurations include the update TM as the primary lookup TM. Please refer to the Translation Memory Administration Guide (TMAG) for more information.

Save

Figure 8: Translation Group Creation

The use of a TM group does not yield the same results as using a single TM that contains all the data of the TMs within a TM group. To effectively use TM groups, you need to understand the order of precedence and how this affects the leverage process. When a TM group is used during the leverage process, it queries each of the TMs in the group individually and in order of precedence.

Important: The key point to remember is that once an acceptable match is found, lower priority TMs will not be leveraged for the current segment.

An acceptable match is defined as either a 100% or ICE match. Within the current TM being leveraged, the best match will be taken. (That is, ICE will always be preferred over 100% matches.) As a result, if a 100% match is found in a TM of higher precedence, then an ICE match candidate for that segment will not be sought in the lower-ranked TMs. (In the same TM, of course, an ICE match is always chosen over a 100% match.)

Split/merge options are also evaluated before moving to the next TM in the TM group. Once a split/merge candidate has been accepted, it cannot be undone. Splitting or merging segments may lead to improved leverage against the current TM. Depending on the configured auto split/merge settings, the resulting segments might not result in a 100% or ICE match. Because the split or merge cannot be undone, the new segments may disrupt the ability to find an acceptable match in TMs of lower precedence. The impact is that the actual results may not be the expected results. Split/merge occurrences can be identified within the Browser Workbench.

Understanding the Use of the Update (or Write) TM

In the configuration of the TM group, the user must specify which TM WorldServer should use to store new or updated translations. Translations are stored in the TM for each asset that is translated. If a new asset is leveraged fully against the applied TM group or if it is newly translated, then new translation entries will be generated for each segment when the asset is saved to the TM. These new TM entries will be stored into the write TM. If the same asset is re-leveraged after being updated, then new TM entries will be generated for the segments that have been newly translated or whose translation has been updated. Therefore, the new and updated translations will be stored in the write TM.

The system recognizes when the TM entry used to leverage a segment is already associated with the asset being leveraged. This is generally the case when working with a new version of a previously translated asset. Because the TM group is treated as a single entity, the default behavior does not consider from which TM the TM entry came, but only that there exists a TM entry in the TM group for the segment and this asset. The fact that the TM entry leveraged for a given segment is already in the TM group with an association to the current asset will prevent it from being written to the write TM, even if it is not explicitly stored in the write TM. This default behavior results in a “leverage it from where it is” type of behavior, and minimizes duplication of TM entries across the TMs within the TM group. All applied or leveraged TM entries will be in the TM group, but will not necessarily end up in the write TM.

Alternatively, the user can configure WorldServer so that the update-TM process will ensure that all saved translations are written to the write TM in the event that the specified write TM does not already contain the applied TM entry. The `save_ice_match_segments` property can be set to

```
true
```

in the `tm.properties` file to ensure that all saved translations end up in the target write TM. (If the TM entry already exists in the write TM, it is not updated unless the translation has been modified.) For information about the `tm.properties` files, see *Configuring TM Properties* on page 144.

TM Precedence Considerations

Ensure that the order of precedence defined for the TM group allows for the desired behavior. The order of the TMs affects the results of the leverage process. As noted earlier, using a TM group is not equivalent to having a single TM with all of the data from the TMs within a TM group.

Using TM Groups

The TM group feature allows a collection of TMs to be accessed during the content leverage process. This section describes how to use TM groups under the following scenarios:

- filtering existing TMs to create a new golden (reference) TM
- protecting an existing golden (reference) TM or TMs
- setting up secondary TMs for leveraging

Creating a Golden or Reference TM

Over time, TMs can grow tremendously. At some point, the user may decide that the current TM contains a lot of “junk” entries. Perhaps a product is no longer supported, or perhaps the segmentation rules have changed, reducing the usefulness of a lot of the data. (Note that segmentation directly affects leveragability against a TM.) For whatever reason, the user may need to filter out some of the data to create a new golden, or reference, TM. To do so, create a TM group as follows:

1. Create a new empty TM to use as the Update TM.
2. Set the reference TMs. Establish the desired order for all the TMs that will be used for leveraging.
3. Set the TM property, `save_ice_match_segments`, to `true`, which forces all leveraged translations to be stored in the write TM.
4. Assign the new TM group to the appropriate AIS path containing the content to be used to populate the new TM. Create a project that will leverage the content and store only the translations that have been fully leveraged. As a result, the new TM will contain entries corresponding to the translated content.
5. Configure the AIS path to use only the new reference TM, and not the group.

Preferring and Protecting a Golden TM

Consider the scenario where you already have a golden TM. This may be some publicly shared TM that has been validated and made available for reference use only. In this scenario, the user wants entries from this TM to have precedence above all others, including TMs with newly added material. The idea may be that for a given project, values might be updated from the entries in the golden TM, but subsequent projects should prefer the golden entries.

In this scenario, configure the TM group in the following manner:

1. Set Update TM to a TM other than the golden TM.
2. Set the reference TMs. Make the Golden TM first, followed by other prioritized TMs.
3. Set the TM property, `save_ice_match_segments`, to `true`, which forces all leveraged translations to be stored in the write TM, depending on whether you want to build the write TM.

As a result of this configuration, during the standard leveraging process, TM entries will always be preferred from the Golden TM.

Leveraging the Old to Supplement the New

In this scenario, it may be desirable to leverage existing TMs (potentially belonging to other groups) to help build out a new preferred TM.

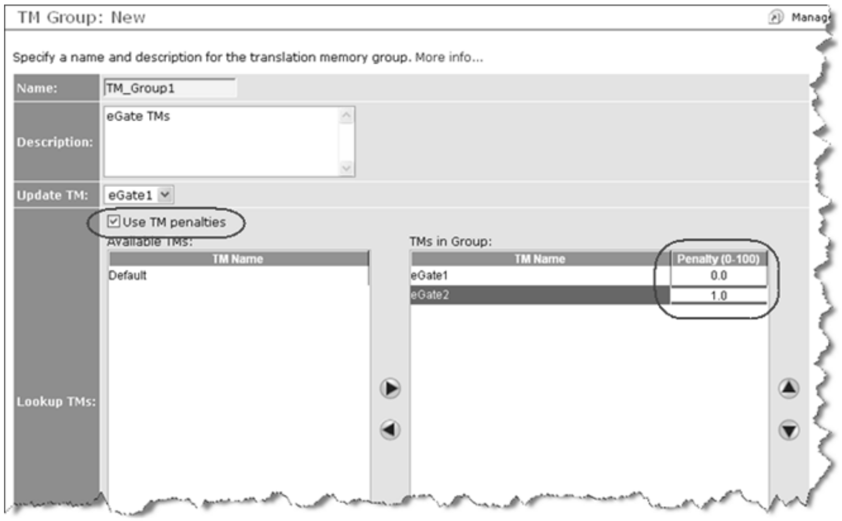
In this scenario, configure the TM group as follows:

1. Set Update TM to the new preferred TM.
2. Set the reference TMs. Set the new preferred TM first, followed by other prioritized TMs.
3. Set the TM property, `save_ice_match_segments`, to `true`, which forces all leveraged translations to be stored in the write TM.

This configuration is similar to the one for creating a new reference TM. However, its continued use differentiates it. Here, the content in the new reference TM is preferred, but the other TMs (which may also continue to be updated via other configurations) are still being leveraged secondarily.

Group Level TM Penalties

When you add a TM to a TM Group, you can define a penalty that will be applied to the matches from that TM.


TM Group: New

Specify a name and description for the translation memory group. More info...

Name:	TM_Group1
Description:	eGate TMs
Update TM:	eGate1
	☑ Use TM penalties

Available TMs:

TM Name
Default

TMs in Group:

TM Name	Penalty (0-100)
eGate1	0.0
eGate2	1.0

Figure 9: TM Group Level Penalties

This penalty is configurable for each TM within the TM group. The **Use TM Penalties** option controls whether or not the assigned penalties are used or not. The TM penalties option is disabled by default. If a penalty is enforced, neither SPICE, ICE, nor exact matches are possible from that TM.

As an example of how you would use the penalties option, if you have a TM for a similar related domain, but want to prevent 100% matches from coming from that TM, you could penalize it.

Alternatively, if you had a TM that contained translations for a slightly different dialect that is similar, but not exactly the same, as the current dialect, you could apply a penalty to ensure that a 100% match never came from that TM.

The penalty represents a leverage level penalty in that the penalty is applied to the match as whole as opposed to being applied against a number, word or placeholder element within the segment. If this penalty is invoked, then the match cannot be repaired up to 100%.

Penalty values must be provided in the range of `0` to `100`. If the penalty is set to `2.0`, for example, the final score is reduced by `2` percentage points. Based on this value, the maximum score that a match against this TM could generate with this penalty is a 98% score.

Note: In the `tm.properties` file, most properties have you enter values in decimal notation (for example, `.02`) while the user interface mostly has you enter values in percent notation (for example, `2`). Therefore, `.02` in the `tm.properties` file means the same thing as `2` in the user interface.

The penalties only apply to the TM group being configured. The penalties associated with the TMs in the group and the apply penalties option are stored with the TM group, not the individual TMs. If the TM is used in a separate TM group, the penalty setting will not carry over from a penalty value set in any other TM group. Additionally, leveraging the TM directly, instead of from a TM group, will not cause penalties for TM matches.

Summary

There are many potential uses for TM groups that extend beyond these examples. However, understanding the above scenarios and configurations should provide sufficient insight to allow you to create TM groups for other purposes.

Chapter

9

Auto Split/Merge

Topics:

The auto-split/merge feature provides the ability to automatically break and combine neighboring segments within an asset to improve overall asset leverage against the applied TM. The need for this feature results from segmentation differences between the asset segments and the segments used to generate the TM entries. Segmentation differences can result for a number of reasons, including users manually splitting or merging segments during the translation process; changes in the filter configuration; importing matches from another TM (third-party or not) whose entries were created under different segmentation rules; and the attempt to leverage translated content across different document types that use different filters.

This chapter describes how to use the auto split/merge feature in WorldServer.

Overview of Auto Split/Merge

The primary purpose for the *auto split/merge* feature is to facilitate the reconstruction of segments that have been manually altered by the user during the translation process. The user can reasonably expect that once a document has been translated in WorldServer, he should be able to completely leverage the same content later against the TM to which the translations were stored. This premise should hold true even if the user manually splits or merges segments during the translation process. The filters are incapable of reconstructing altered segments because such segments, by definition, are inconsistent with the filters segmentation strategy.

Even though the auto split/merge feature has been created to address segmentation differences that result from user-defined segments, it can also handle generic segmentation differences that may occur as a result of variations in segmentation across different filters and the TMs that have been created.

The core concept of the auto split/merge feature is that WorldServer tries to adjust the segmentation for previously split or merged segments, based on the TM matches it finds during the TM leverage process. During this process, TM match candidates are found and processed in the order in which they are retrieved. SP/ICE, exact and 100% match candidates for the segment are not guaranteed. The match or matches that are retrieved may not even be high scoring matches. In the event that SP/ICE, exact and 100% matches are not found, the system will subject the matches to the auto-split/merge process. The process applies the auto-split/merge sub-processes in the following order:

1. **standard auto-merge**

   Checks to see if the match candidate contains the current asset segment followed by the next asset segment. This process is extremely sensitive to formatting (placeholder) differences.
2. **hyper-merge**

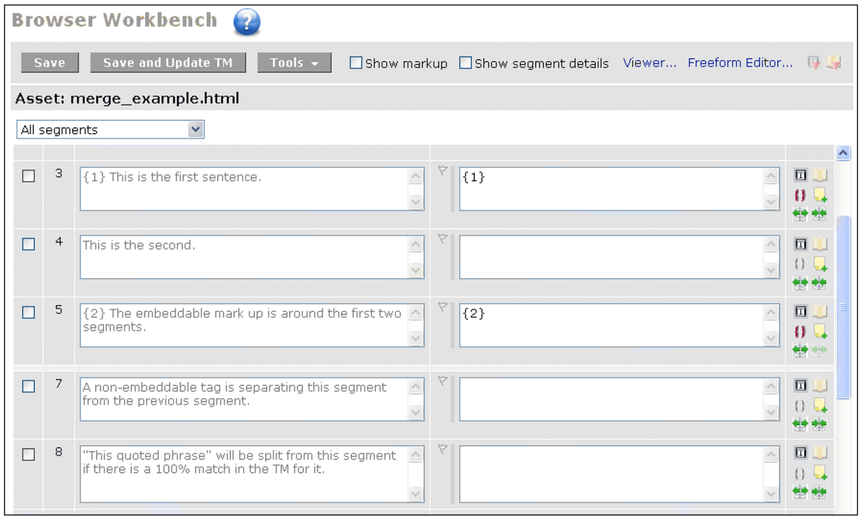
   Same as the standard auto-merge, except that it ignores formatting differences.
3. **auto-split**

   Checks to see if the current match candidate represents the beginning text of the current asset segment. This process is extremely sensitive to formatting (placeholder) differences.

The auto-split process asserts that if a more complete match cannot be found, it is preferable to accept a 100% match, or better, for part of the segment, than to accept a lower overall score for the complete original segment. The merge process asserts a similar argument — that it is preferable to accept a higher collective score for a larger segment, than to accept lesser scores for multiple smaller segments. The remainder of this section provides details on each component of this technology.

Auto-Merge

The auto-merge functionality is pretty simple. Consider the following asset portion:



Browser Workbench

Save | Save and Update TM | Tools | Show markup | Show segment details | Viewer... | Freeform Editor...

Asset: merge_example.html

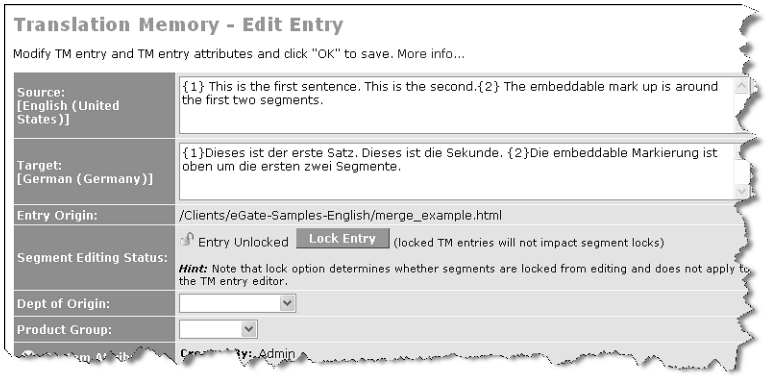
All segments

	#	Source	Target
☐	3	{1} This is the first sentence.	{1}
☐	4	This is the second.	
☐	5	{2} The embeddable mark up is around the first two segments.	{2}
☐	7	A non-embeddable tag is separating this segment from the previous segment.	
☐	8	"This quoted phrase" will be split from this segment if there is a 100% match in the TM for it.	

Figure 10: Browser Workbench Segmentation Example 1

The preceding screen shot represents the segmentation generated by the applied filter.

Consider the following TM entry.



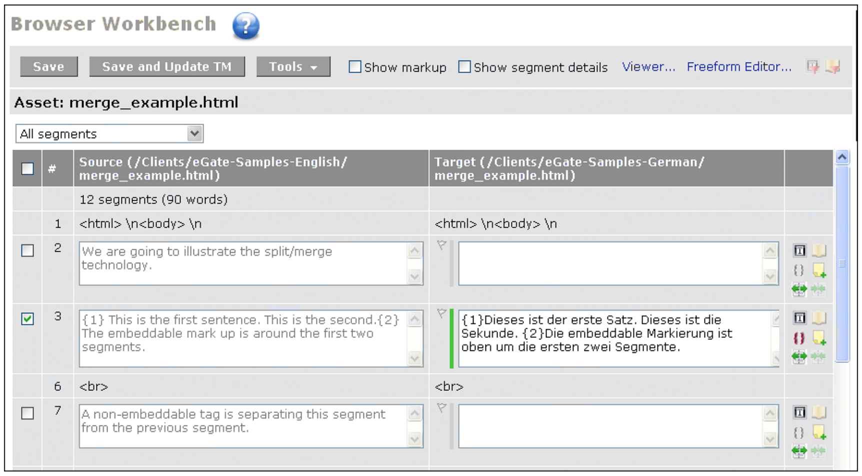
Translation Memory - Edit Entry

Modify TM entry and TM entry attributes and click "OK" to save. More info...

Source: [English (United States)]	{1} This is the first sentence. This is the second.{2} The embeddable mark up is around the first two segments.
Target: [German (Germany)]	{1}Dieses ist der erste Satz. Dieses ist die Sekunde. {2}Die embeddable Markierung ist oben um die ersten zwei Segmente.
Entry Origin:	/Clients/eGate-Samples-English/merge_example.html
Segment Editing Status:	Entry Unlocked Lock Entry (locked TM entries will not impact segment locks) *Hint:* Note that lock option determines whether segments are locked from editing and does not apply to the TM entry editor.
Dept of Origin:	
Product Group:	
	Admin

Figure 11: Translation Memory Entry Editor - Example 1

If this entry is saved into the TM, and the above asset is re-leveraged against the TM, the following will result.



Browser Workbench

Save | Save and Update TM | Tools | Show markup | Show segment details | Viewer... | Freeform Editor...

Asset: merge_example.html

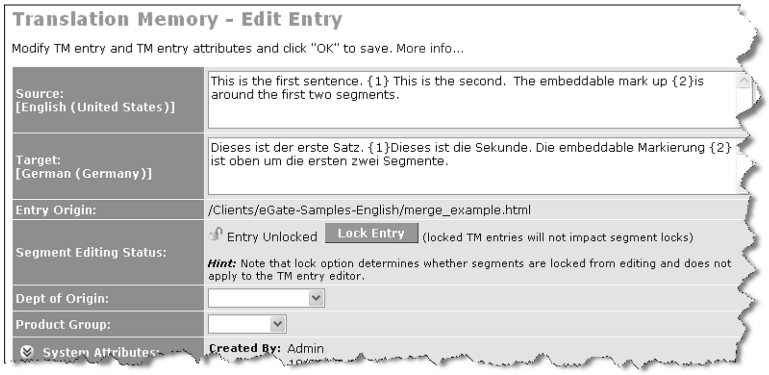
All segments

☐	#	Source (/Clients/eGate-Samples-English/merge_example.html)	Target (/Clients/eGate-Samples-German/merge_example.html)
		12 segments (90 words)	
	1	<html> \n<body> \n	<html> \n<body> \n
☐	2	We are going to illustrate the split/merge technology.	
☑	3	{1} This is the first sentence. This is the second.{2} The embeddable mark up is around the first two segments.	{1}Dieses ist der erste Satz. Dieses ist die Sekunde. {2}Die embeddable Markierung ist oben um die ersten zwei Segmente.
	6	 	
☐	7	A non-embeddable tag is separating this segment from the previous segment.	

Figure 12: Browser Workbench Segmentation Example 2

The two segments have been merged together to improve leverage against the match. Merging and translating these two segments together in another asset resulted in this TM entry being created. This merged segment was created manually, and the auto-merge process was able to effectively leverage against the merged TM entry.

The auto-merge process is highly sensitive to formatting (placeholder) differences. The addition of any new formatting data will invalidate the TM entry as an auto-merge candidate. For example, the following TM entry will fail the condition for this merge option:



Translation Memory - Edit Entry

Modify TM entry and TM entry attributes and click "OK" to save. More info...

Source: [English (United States)]	This is the first sentence. {1} This is the second. The embeddable mark up {2}is around the first two segments.
Target: [German (Germany)]	Dieses ist der erste Satz. {1}Dieses ist die Sekunde. Die embeddable Markierung {2} ist oben um die ersten zwei Segmente.
Entry Origin:	/Clients/eGate-Samples-English/merge_example.html
Segment Editing Status:	Entry Unlocked Lock Entry (locked TM entries will not impact segment locks) *Hint:* Note that lock option determines whether segments are locked from editing and does not apply to the TM entry editor.
Dept of Origin:	
Product Group:	
System Attributes:	Created By: Admin

Figure 13: Translation Memory Entry Editor - Example 2

The difference in positions of the placeholders, is enough to cause the failure. The auto-merge is in effect an additive exact match merge. The merged pieces exactly match portions of the TM match.

A merge is generally identifiable in the browser workbench by a discontinuity in the segmentation numbers. Before the merge, the segment IDs are in numerical order. (Often you have to select the **show markup** options to fully see that the IDs increment in order. Refer back to the original asset illustration.) After a merge, all segment IDs between the first segment and the last segment involved in the merge (including the ID of the last segment) are lost. Notice in the most recent screenshot of the Browser Workbench that text segment #4 no longer exists (see the second column).

This example has illustrated two segments being merged together. The auto-merge process can merge an indefinite number of segments to match up to a super TM entry, provided that no complete matches are found along the way.

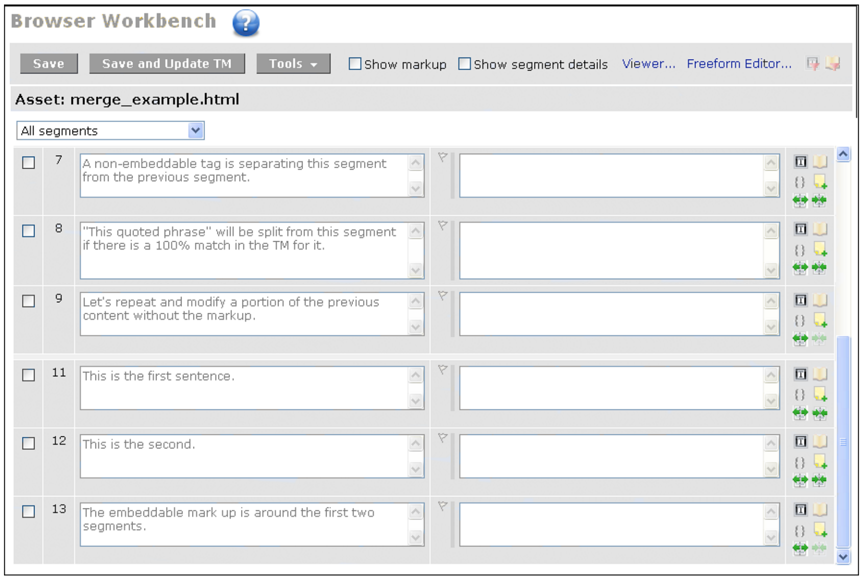
Note: Merging is not allowed across markup boundaries.

Hyper-Merge

Hyper-merging evaluates merge candidates based on language content alone. Placeholders are ignored initially, but are still used to evaluate final scoring of merged segments. Depending on the assigned placeholder penalties, the final segment may result in additional 100% matches, but most probably will lead to higher fuzzy matches only.[5]

For example, consider the earlier scenario where the first segment has been modified, as shown in the following figure.

[5] 100% matches generally occur only when the placeholder penalties have been set to zero.



Browser Workbench

Save | Save and Update TM | Tools | Show markup | Show segment details | Viewer... | Freeform Editor...

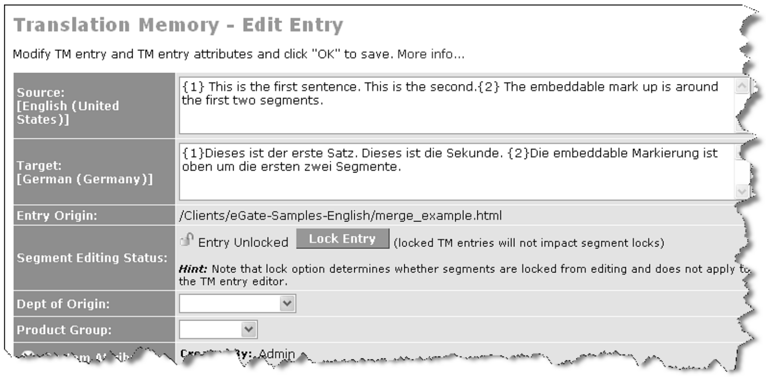
Asset: merge_example.html

All segments

7	A non-embeddable tag is separating this segment from the previous segment.	
8	"This quoted phrase" will be split from this segment if there is a 100% match in the TM for it.	
9	Let's repeat and modify a portion of the previous content without the markup.	
11	This is the first sentence.	
12	This is the second.	
13	The embeddable mark up is around the first two segments.	

Figure 14: Browser Workbench Segmentation Example 3

Next, seed the TM with the following TM entry (which is the same entry as we used earlier):



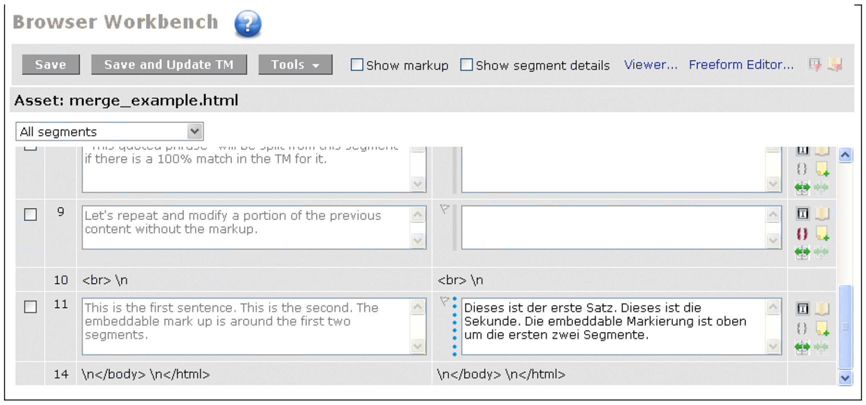
Translation Memory - Edit Entry

Modify TM entry and TM entry attributes and click "OK" to save. More info...

Source: [English (United States)]	{1} This is the first sentence. This is the second.{2} The embeddable mark up is around the first two segments.
Target: [German (Germany)]	{1}Dieses ist der erste Satz. Dieses ist die Sekunde. {2}Die embeddable Markierung ist oben um die ersten zwei Segmente.
Entry Origin:	/Clients/eGate-Samples-English/merge_example.html
Segment Editing Status:	Entry Unlocked Lock Entry (locked TM entries will not impact segment locks) *Hint:* Note that lock option determines whether segments are locked from editing and does not apply to the TM entry editor.
Dept of Origin:	
Product Group:	

Figure 15: Translation Memory Entry Editor - Example 3

When we reapply the TM to this asset, we get the following segmentation.



Browser Workbench

Save | Save and Update TM | Tools | Show markup | Show segment details | Viewer... | Freeform Editor...

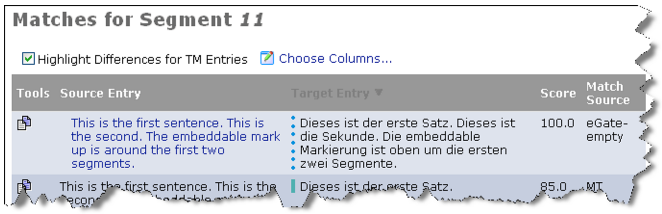
Asset: merge_example.html

All segments

	"This quoted phrase" will be split from this segment if there is a 100% match in the TM for it.	
9	Let's repeat and modify a portion of the previous content without the markup.	
10	 \n	 \n
11	This is the first sentence. This is the second. The embeddable mark up is around the first two segments.	Dieses ist der erste Satz. Dieses ist die Sekunde. Die embeddable Markierung ist oben um die ersten zwei Segmente.
14	\n</body> \n</html>	\n</body> \n</html>

Figure 16: Browser Workbench Segmentation Example 4

Even though the translation has not been populated in the target segment, the source segments have still been merged. Clicking the TM entries button shows the following match.



Matches for Segment 11

☑ Highlight Differences for TM Entries   Choose Columns...

Tools	Source Entry	Target Entry	Score	Match Source
	This is the first sentence. This is the second. The embeddable mark up is around the first two segments.	Dieses ist der erste Satz. Dieses ist die Sekunde. Die embeddable Markierung ist oben um die ersten zwei Segmente.	100.0	eGate-empty
	This is the first sentence. This is the	Dieses ist der erste Satz.	85.0	MT

Figure 17: Translation Memory Entry Editor - Example 4

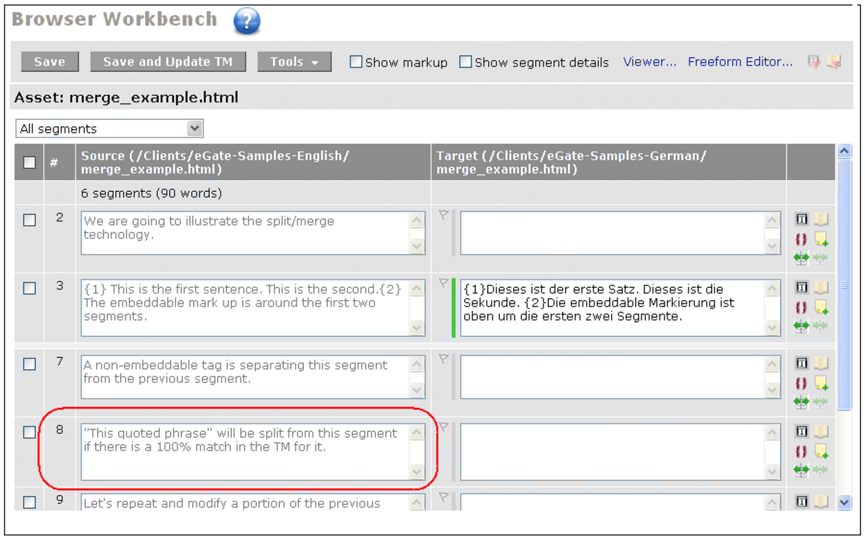
In this example, the final score is a 100% match due to the safe placeholder repair. However, often the score for the hyper merged match is less than 100%. This is still beneficial in that instead of two segments with low fuzzy matches, there is a single segment that yields a high fuzzy match. This result has significant implications to final translation cost that are geared toward varying fuzzy ranges.

The above example shows two segments being merged together. The hyper-merge process can merge an indefinite number of segments to match up to a super TM entry, provided that no complete matches are found along the way.

Auto-Split

The auto-split process attempts to fully leverage a portion of the segment in the event that the segment cannot be fully leveraged as-is. If the match exactly represents the first part of the segment, it will split the original segment at that point, and create a second segment from the remainder of the original segment. The TM match will then be applied to the leading segment, and the trailing segment will be leveraged on its own.

Consider the following asset.



Browser Workbench

Save | Save and Update TM | Tools | ☐ Show markup | ☐ Show segment details | Viewer... | Freeform Editor...

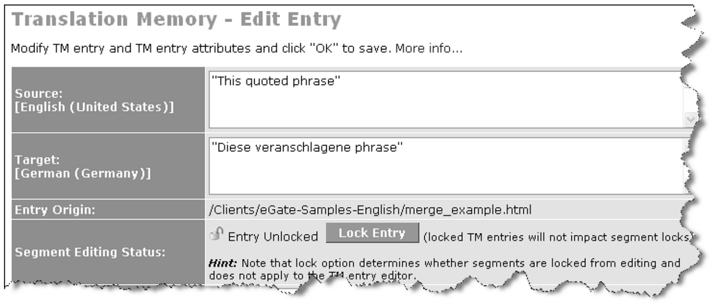
Asset: merge_example.html

All segments

☐	#	Source (/Clients/eGate-Samples-English/merge_example.html)	Target (/Clients/eGate-Samples-German/merge_example.html)
		6 segments (90 words)	
☐	2	We are going to illustrate the split/merge technology.	
☐	3	{1} This is the first sentence. This is the second.{2} The embeddable mark up is around the first two segments.	{1}Dieses ist der erste Satz. Dieses ist die Sekunde. {2}Die embeddable Markierung ist oben um die ersten zwei Segmente.
☐	7	A non-embeddable tag is separating this segment from the previous segment.	
☐	8	"This quoted phrase" will be split from this segment if there is a 100% match in the TM for it.	
☐	9	Let's repeat and modify a portion of the previous	

Figure 18: Browser Workbench Segmentation Example 5

If the TM is seeded with the following entry, a split condition will be generated for the asset.

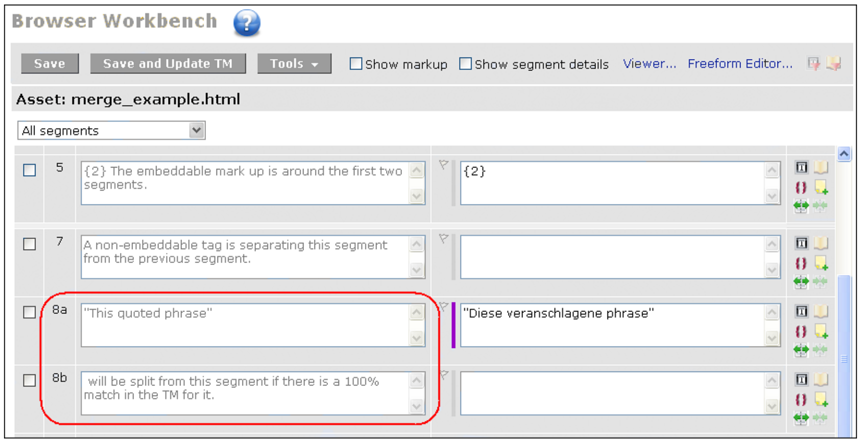

Translation Memory - Edit Entry

Modify TM entry and TM entry attributes and click "OK" to save. More info...

Source: [English (United States)]	"This quoted phrase"
Target: [German (Germany)]	"Diese veranschlagene phrase"
Entry Origin:	/Clients/eGate-Samples-English/merge_example.html
Segment Editing Status:	Entry Unlocked Lock Entry (locked TM entries will not impact segment locks) *Hint:* Note that lock option determines whether segments are locked from editing and does not apply to the TM entry editor.

Figure 19: Translation Memory Entry Editor - Example 5

If the asset is re-leveraged against the TM, the following segmentation changes will result from the auto-split condition.



Asset: merge_example.html

5	{2} The embeddable mark up is around the first two segments.	{2}
7	A non-embeddable tag is separating this segment from the previous segment.	
8a	"This quoted phrase"	"Diese veranschlagene phrase"
8b	will be split from this segment if there is a 100% match in the TM for it.	

Figure 20: Browser Workbench Segmentation Example 6

The first part yielded a 100% match (which will always be the case.) However, there is no guarantee for the type of match that will be found for the second part of the original segment.

Spotting a split condition is even easier than spotting a merge condition. As in this example, the resulting segments always carry the ID of the original segment, followed by an order letter.

When to Use Auto-Split/Merge

The auto-split/merge feature can be optionally used within the WorldServer environment. By default, it is partially enabled. The merge processes within the auto-split/merge feature are generally deemed more reliable than the auto-split process. When an appropriate merge candidate is found, there is a guarantee that both asset segments involved will be leveraged higher against the applied TM than they would be on their own. (There is no guarantee that the larger merged segment will lead to a 100% or better match.) For auto-splits, the only known quantity is that the first part of the segment will result in a 100% match, while the new segment created from the second part of the original segment may not even have a TM match in the applied TM.

There are two processes during which the auto-split/merge process is generally applied: ICE lookup and fuzzy lookup. If translators are expected to manually split or merge segments during translations, then the ICE matches that would be expected after re-leverage on the same content depend on whether the split/merge options for the ICE lookup are enabled. For this reason, the options for the ICE lookup process are enabled by default. The hyper-merge option does not apply to the ICE match lookup process. Both merge options are enabled for the fuzzy lookup process by default. The auto-split option is disabled by default for the fuzzy lookup process.[6]

[6] The auto-split process can be useful in the fuzzy process. However, it is highly dependent on the characteristics of customer data. There have been situations where the split functionality has led to less than desirable results. However,

In general, the split/merge technology can be effectively used when significant segmentation differences exist between the asset and the TM. For instance, consider using the split technology when the TM mostly contains sentence-level entries and the asset filter produces paragraph-level segments. The split process can potentially break up the asset segments to make better matches against the TM entries.

Similarly, the merge process can be effectively used when the asset segments are produced at the sentence-level, but the TM entries were created at the paragraph-level. The merge process can progressively merge successive asset segments to match against a larger TM entry.

Both technologies can be used in tandem to leverage against a TM that may contain both sentence and paragraph-level entries. It is worth restating that this technology is only employed in the event that a qualifying match cannot be found for the original segment.

Split/Merge Technology Can Mask Segmentation Differences

The automatic split/merge technology provides a way to overcome segmentation differences to improve leveragability of TM data. The technology works as a segment adjuster. It modifies the asset's segmentation to enhance the chances of finding a match in the TM. Segments that have been updated by the split/merge technology will tend to match the segmentation of the historical TM entries. Future matches for the same content will have to go through the same process each time, which has a performance cost.

Impact of TM Groups and Auto-Split/Merge

The auto-split merge feature is geared toward maximizing the leverage against the current TM being applied. This does not change even when the user is using TM groups. As a result, the leverage levels against secondary TMs in the TM group may not be as expected because auto-split/merge candidates may be found in the current TM being leveraged. Once a split or merge is done, it cannot be undone for the next TM.

Configurable TM Properties

The auto-split options can be configured. They are exposed as TM properties. The properties that apply to the split merge process are as follows:

- `do_fuzzy_automatic_merge` (`True` to enable, `false` to disable; enabled by default)
- `do_fuzzy_automatic_split` (`True` to enable, `false` to disable; disabled by default)
- `do_hyper_merge` (`True` to enable, `false` to disable; enabled by default — does not apply to ICE process)
- `do_ice_automatic_merge` (`True` to enable, `false` to disable; enabled by default)
- `do_ice_automatic_split` (`True` to enable, `false` to disable; enabled by default)

Refer to the Appendix for more details on each option and how to set each option.

there have also been significant innovations to the implementation to address many and perhaps all of these past issues. Users need to evaluate the impact of the fuzzy split option on their data.

Chapter 10

Path Normalization: Sharing AIS Context Among Assets

Topics:

WorldServer supports sharing TM AIS context across multiple assets. This chapter explains why and how you would use this feature.

Overview of TM AIS Path Normalization

During the storage of translation pairs, the TM entries are associated with the TM AIS context of the asset from which they were created. Each asset effectively has its own set of translations within the TM. If the asset is updated and retranslated, the new translations will replace the previously stored translations that are associated with the asset. This allows WorldServer to maintain separate translation sets for similar or identical content shared across various assets. For example, if two assets with identical content are translated differently, the TM will contain a separate set of translations for each asset. When re-leveraging an asset against a TM, WorldServer will prefer TM entries that were created from an earlier version of that asset, even if there are newer TM entries from another asset for the same content. (See the section on WorldServer Rules for Ranking Matches.) This is important in that it allows WorldServer to regenerate target files long after they were originally created.

There have been occasions where customers have desired that the TM AIS context be shared between multiple assets. For example, a customer may be working on different versions of the same file, and the different versions may be stored at different locations within the AIS mounts. When translated, each of these versions will have their own set of translations in that they are treated as separate assets.

Path normalization is the process through which the TM AIS context can be modified in order to support the sharing of the TM AIS context between multiple assets. TM AIS context, similar to the AIS context[7], is based on the AIS path of the asset being translated. The path normalization facilities allow you to augment the TM's definition of AIS path. For example, AIS path differences related to version differences can be normalized out of the AIS paths.

Consider the following AIS paths assets:

```
/AIS/VERSION1/EN/PRODUCT1/introduction.doc
/AIS/VERSION2/EN/PRODUCT1/introduction.doc
```

The above referenced assets may contain different versions. WorldServer will treat these as two separate assets because the AIS paths are different and because WorldServer does not require asset names to be unique. If the customer wanted these two assets to share the same TM translations, then the path normalization process would need to be overridden to negate the differences between the two paths. One normalization strategy might be to simply remove the version element from the two paths:

```
/AIS/EN/PRODUCT1/introduction.doc
/AIS/EN/PRODUCT1/introduction.doc
```

Now, the two paths will look identical when presented to the TM storage level, and the TM will see both assets as being the same asset.

What does it mean for two assets to have the same TM AIS context?

- They share a single set of TM translations.
- The translations in the TM will match those of the most recently translated version.
- Potential loss of translation regeneration capability of all assets. If particular versions of the assets sharing the TM AIS context are not identical in content, the TM entries will only completely match the segments of the last translated version. This means that it is possible to lose some of the associated translations or that some ICE matches may become merely exact matches due to potential content and segmentation differences between the different versions. Re-leveraging a previously translated version may no longer be completely ICE matched or translatable against the TM.[8]

Updates to one asset will affect the translations for all other assets sharing the TM AIS context when they are re-leveraged.

[7] In most cases, the TM AIS context and AIS context are identical. They only differ when the path normalization process is utilized.

[8] Translations of earlier versions of the asset can be regenerated reliably provided the previous asset segmentation information has not been invalidated. The path normalization process affects how the TM identifies assets, but it does not affect how the AIS identifies assets. If an asset has not been changed, deleted or had its cached cleared, then the AIS can generate the target files based purely on the translations stored with the segmentation data. This process will not even go to the TM.

Overriding the Path Normalization Implementation

The path normalization process can be overridden by providing a new implementation for the exposed TM path normalization service. This SDK supported component allows you to provide your own path normalization implementation. See the SDK documentation for details on the TM services component (`WSTmServicesComponent`).

Note: Note that path normalization implementations need to be reentrant or repeatable. The path normalization process is called when entries are stored to the TM, which happens during an update TM operation and a TM import operation. Running during the normal translation process is safe as it always starts with the asset AIS path. However, TM entries that are exported may or may not already have the normalized AIS path. As a result, the process will run when importing TMX files as well. This is why the process should be reentrant.

Dangers and Effects of Using Path Normalization

In general, SDL suggests that you do not use this functionality in that it can be problematic if not fully understood and implemented correctly. However, if you have requirements that this feature would allow you to meet, then carefully consider the scenarios described below. Improper use of this functionality can be very costly. You are encouraged to consult WorldServer Professional Services before pursuing this option.

The Ping-Pong Effect

The ping-pong effect describes the behavior of the TM when customers seek both to employ the path normalization technology to relate various exposed versions of an asset, and to employ the basic translation regeneration capability of WorldServer. As described earlier, WorldServer TM technology allows customers to regenerate translations for past work long after substantial updates have been made to the data stored in the TM. This ability is based on storing translation sets associated with each asset within the TM. Within the TM, an asset is identified by the associated TM AIS context. As long as the TM AIS context does not change and is not shared, the customer can reliably regenerate the translation of past-translated work. If the asset changes, the system can reliably translate the unchanged content identically to the previous translation. This ability is directly dependent on each asset having its own set of translations within the TM.

The problem is that these capabilities are mutually exclusive. It will not always be possible to regenerate a previous translation version when using path normalization. Path normalization results in multiple assets sharing a single set of TM entries. It will cause TM entries for all versions except the most current versions to be expunged from the TM. The TM will only contain TM entries associated with the last translated asset for any given TM AIS context.

Note that this problem only occurs (practically) in the event that two or more assets sharing a TM AIS context are being updated and translated over an overlapping period of time. For example, multiple releases may be authored for the “same asset”. The content may change for both over a given amount of time, and as they change, they are iteratively translated. If these assets are forced to share the same TM AIS context, each translation round will negatively affect the translation of the other. This back-and-forth behavior lends itself to the ping-pong effect name. This is illustrated in the following example.

Example

Let's assume we are working on Release #1 and that in a few days we will need to work on Release #2 before Release #1 has been released.

On Release #1 we translate a file (A). All segments are committed to TM so that the next time the file is parsed all segments are ICEd.

The file segments look like this:

```
Seg. 1: This is segment 1
Seg. 2: This is segment 2
Seg. 3: This is segment 3
```

A few days later Release #2 is ready to be translated. We translate file (A), which is updated with a few new segments:

```
Seg. 1: This is segment 1
Seg 4: This is a new segment 4
Seg. 2: This is segment 2
Seg 5: This is a new segment 5
Seg. 3: This is segment 3
```

The Linguist will translate the file and save it to the TM so that all strings are now ICE'd against the TM.

Just before finalizing Release #1, file (A) gets a fix on the code level that does not influence the value or any of the segmentation rules. The file will be submitted for translation using a WorldServer project. The workflow being used will auto generate the target file provided that the asset is all ICE’d.

Results

However, file (A) no longer registers all ICE’d segments. The file will stop with 100% matches instead of ICE matches. If the Engineer or the Linguist commits the translation to memory, then the same file (A) on Release #2 will have its values changed to 100% when leveraged against the TM, instead of being completely ICE'd.[9] If the TM is updated again with the translations from file (A) on Release #2, then the matches for file (A) on Release #1 will revert back to 100% when re-leveraged against the TM.

Over Normalization and Accidental Loss of Translations

Over normalization is the result of defining path normalization rules that cause too many assets to share a common TM AIS context. The worst case occurs when all AIS paths resolve to the same TM AIS context. In effect, the TM will see all assets as being the same asset, and only one set of translations will be stored. While this will keep your TM from ever getting too large, it will also ensure you that your TM will not grow in value.

The nature of path normalization is one that results in reducing TM entries. The normalization algorithms used must be rigorously tested and validated prior to being used in production.

[9] Note that this will happen if the file has not changed and it is forced to be re-leveraged by triggering segmentation. If file (A) in Release #2 was completely translated before additional work is done on file (A) in Release #1, then WorldServer would be able to generate the previously translated file. However, the slightest change to file (A) in Release #2 would result in the ping-pong effect.

Chapter 11

Automatic Translation Configuration

Topics:

WorldServer provides auto-translation options for dealing with content that typically would remain the same for both the source and target. For example, symbols and codes are entities that do not require translation, or rather, are translated as-is. Instead of requiring and paying translators to do these auto-translations, WorldServer can perform them automatically. This chapter describes the default support that WorldServer offers and the conditions under which each feature applies.

You can extend or even replace the auto translation support by using the TM Services Framework, as documented and supported in the WorldServer SDK, introduced in WorldServer Version 7.5. The description that follows details the default implementation that is packaged and enabled with WorldServer.

Rules for Auto Translating Segments

In the default implementation, WorldServer attempts to address common scenarios that in most cases do not require a language-specific translation. The cases are as follows:

Rule 1	**labels** – segments that contain only words with n or fewer characters, and potentially numbers and placeholders; ("n" is a specified number). An example might be "1 NW 50"
Rule 2	**number-only segments** – segments containing a number only, and optionally, placeholders
Rule 3	**content variables** – segments with combinations of numbers and letters without spaces, and having only the allowed separator characters, which are a part of the variable; placeholders are allowed in the segment, but only at the ends. An example might be the notation for a serial number, such as SN:213108124

The segments that satisfy the auto-translate conditions are copied exactly from source to target. This happens during the TM leverage process after ICE and exact lookups have failed to retrieve a match.

Note: These rules are applied to the entire segment, not to words or fragments of the segment.

Scoring and Status of Auto-translated Segments

During the leverage process, segments that are auto-translated receive a 100% score and are marked as machine translated. For scoping purposes, these segments represent additional 100% matches. Auto-translated segments never register ICE matches until the translation has been saved to the TM.

Auto-Translation Configuration Options

The use of the auto-translate functionality can be configured by setting properties in the `tm.properties` file as illustrated by the following sample `tm.properties` excerpt:

```
# define the minimal size word; if all words in a segment are
# less than this size, then the entire segment will be auto-translated
min_letters_in_word=3
#
# to enable auto-translations
enable_auto_translate=true
#
# translate variables (like SKUs)
auto_translate_variables=true
# define mid characters for variables.
# I.e., to support SN:123-45-6789 ABDC-:12323
# set auto_translate_mid_variable_markers=:-
# default value allows no mid characters, i.e., ABDS123345
auto_translate_mid_variable_markers=
```

Note: The default for `min_letters_in_word` changed in WorldServer 9.0 from `3` to `0` to accommodate Asian languages like Japanese and Chinese, which all have one-character words. For these languages, if this value were `3`, all words would be auto-translated.

Default Implementation Samples

The following table shows the result of these settings on example segments, where `n = 3`, and no mid-characters are allowed for content variables.

Example Segment	**Is Auto-Translated?**	**Reason/Rule**[10]	**Notes**
Abc	Yes	#1	
Hat	Yes	#1	Use of this option can result in undesirable translations. Set the value of n to a low number to minimize this effect.
ABRR.	No		Word contains more than n characters.
(617) 123-4567	Yes	#2	
{1} 234#, {2} 234.453	Yes	#2	
AB. 435 {1}-43	Yes	#1	
AB1234	Yes	#1, 3	
AB{1}1234	Yes	#1	Rule #3 fails because of the placeholder.
ABCD1234	Yes	#3	
ABCD{1}1234	No		Rule #3 fails because of the placeholder.

[10] **See the earlier section, "Rules for Auto Translating Segments".**

Chapter 12

TM Entry Attributes

Topics:

Each TM entry that is added to the TM contains attributes that describe some aspect of the TM entry. For instance, the creation date tells you when the entry was created. The last modified user tells you which user is responsible for the last update for this entry. In addition, WorldServer allows you to define attributes that will be associated with each TM entry. The ability to create these custom attributes allows you to associate additional information that can provide additional insight to the nature of the entry. For instance, if you wanted to track the business group and product for which that TM entry was generated, you might create a "business group" and "product" TM (translation) entry attribute.

The details of how to create these attributes are provided in the *WorldServer User Guide*. This chapter describes how to populate TM entry attributes and how to use them.

Populating TM Entry Attributes

Once you have created the TM entry attribute definitions, you will need to decide how you will populate these values. WorldServer will always populate system level attributes such as creation date, creation user, modified date and modified user. However, you are responsible for defining how custom attributes are to be populated. There are five primary ways of populating TM entry attributes:

- Manually
- Importing attribute values from a TMX file
- Programmatically
- Automatically mapping values from AIS properties
- Automatically mapping values from segment attributes

Manually Editing TM Entry Attributes

WorldServer provides a TM entry editor that presents all of the TM entry data which includes source and target text, system attributes, and all defined custom TM entry attributes. The TM entry editor can be accessed from a number of locations depending on the user's privileges. Generally, the WorldServer search or lookup page that displays TM matches provides a link to the TM entry editor for a given match.

Within the TM entry editor, you are allowed to modify the source and target text, any custom attribute values, and any system attribute that is not exclusively controlled by the system. For instance, you would be able to set the lock status on a TM entry.

While this mode of assigning and editing attribute values is supported, it is the slowest and most error prone approach.

Importing Attributes from a TMX File

TMX files from an existing WorldServer environment or from another vendor may already contain attribute values that you want to migrate into WorldServer.

Programmatically Populating TM Entry Attributes

The WorldServer SDK provides support for accessing and updating TM entry attribute values. If the values to be populated are patterned or can be described heuristically, the SDK provides a means for defining an automated process for populating TM entry attributes.

For example, suppose you wanted to combine multiple TMs from different projects into a single TM, but you wanted to identify the project origin of the various TM entries. You could create a "Project" attribute, and then programmatically assign the project value to each of the TM entries. You could do this in two basic ways:

- • **Update TM entries based on TM search –** This method requires that you be able to identify the TM entries using the SDK TM search APIs. You would then programmatically update each TM entry as appropriate.
- • **Add TM entry attribute information to TMX import** – As described earlier, attribute information stored in the TMX file will be imported into the new TM entries, provided the corresponding attribute exists within WorldServer. In this solution, you could programmatically or manually modify an existing TMX file so that it contained the TM entry attribute values you want to set. Importing the TMX will automatically set these values in the TM entries created in the import job.

Mapping AIS Properties To TM Attributes

In WorldServer, you can assign properties to assets either to associate WorldServer objects with the assets or to further describe the nature of the assets. Typically, you assign properties to a group of assets, but you can also assign properties to an individual asset. Generally, the assigned properties revolve around standard WorldServer constructs. For example, you can assign a base TM, TD, or Workflow to an entire collection of assets that share a common AIS path. By default, all assets beneath that path inherit the properties assigned to the base path. WorldServer also allows users to create their own properties, based on their internal business requirements.[11]

WorldServer also allows you to create custom TM attributes, allowing customer-specific data to be stored along with TM entries. These values are populated during or after the translation process. WorldServer automatically populates standard TM attributes (such as creation date and user). However, to store values in custom attributes, you must write custom processes. Some customers use custom automatic actions to provide values for the TM entries during project execution. Some customers populate many of these values during the TM import process, although this does not address future TM entries that are generated during the asset translation process.

As an alternative way to populate custom TM entries, WorldServer allows you to map custom AIS text or selector properties to custom TM text or selector properties. If you enable this property mapping, the value of the mapped AIS property will be stored in the corresponding TM attribute for any new TM entries created during the translation process, as long as a value has been set for the asset being translated. For instance, if there is an AIS property called “product” that is automatically mapped to a TM attribute named “product,” then the value assigned to the asset being translated will be stored in every TM entry created by translations from that asset. If there are multiple mapped AIS properties, each will be mapped accordingly.

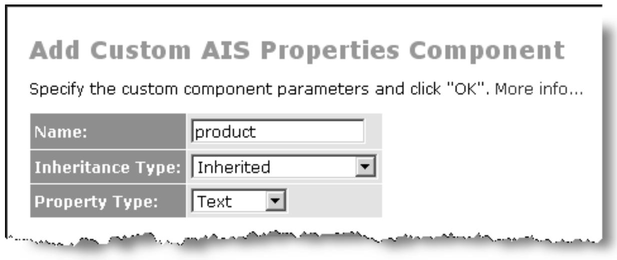
Note: Source (not target) AIS properties are mapped to TM attributes.

How to Map AIS Properties to TM Attributes

The mapping process is simple: AIS properties and TM attributes are mapped based on their names. If the AIS property name matches identically that of a TM attribute’s internal (non-UI) name, then these two are automatically mapped.

The steps for mapping AIS properties to TM attributes are summarized in the following steps:

1. Create a new AIS property:
   a) Go to the Customization page, and add a new AIS property.
   b) Set the inheritance type to `Inherited`.
   c) Set the property type to either `Text` or `Selector`.
   d) TM attributes can be text values only, so the AIS properties mapped to them should be text as well. You can either allow users to enter any text value, or restrict them to choosing a text value from a selector list.
   e) For example:



Add Custom AIS Properties Component

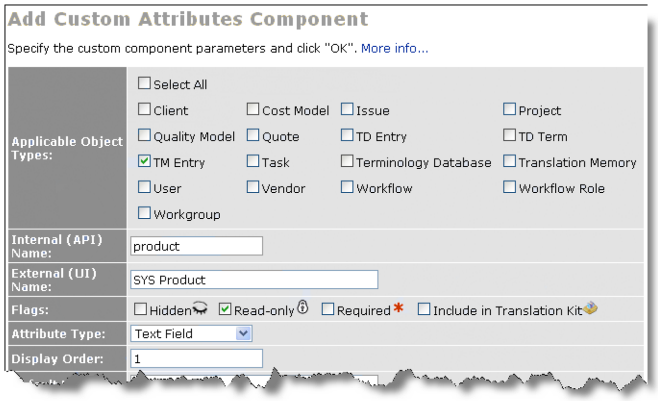
Specify the custom component parameters and click "OK". More info...

Name:	product
Inheritance Type:	Inherited
Property Type:	Text

2. Create a TM attribute:
   a) Go to the WorldServer Customization page, and add a new attribute.
   b) For Applicable Object Type, select `TM Entry`.
   c) Set the Internal (API) Name to match the name for the AIS property.
   d) Set the attribute type to `Text Field` or `Selector`.
   e) Set the flags based on desired exposure. Select `Hidden` to prevent users from seeing this value when reviewing TM entries. Select `Read-only` to prevent users from updating the value.

[11] WorldServer allows you to determine how assets in subdirectories of the AIS inherit properties. The creation and administration of AIS properties are outside of the scope of this document.

f) For example:


Add Custom Attributes Component

Specify the custom component parameters and click "OK". More info...

Applicable Object Types:	☐ Select All ☐ Client ☐ Cost Model ☐ Issue ☐ Project ☐ Quality Model ☐ Quote ☐ TD Entry ☐ TD Term ☑ TM Entry ☐ Task ☐ Terminology Database ☐ Translation Memory ☐ User ☐ Vendor ☐ Workflow ☐ Workflow Role ☐ Workgroup
Internal (API) Name:	product
External (UI) Name:	SYS Product
Flags:	☐ Hidden ☑ Read-only ☐ Required * ☐ Include in Translation Kit
Attribute Type:	Text Field
Display Order:	1

3. Assign the AIS property to assets, directories, or both:

a) Select the asset or asset path and assign the desired value for the AIS property.

b) Repeat for all assets and asset paths, as required.

After you have completed these steps, the TM engine will pull the value from the asset being translated and use that value to populate the mapped TM attribute field for newly created TM entries.

Enabling or Disabling the Mapping from AIS Property to TM Attribute

The ability to map AIS properties to TM attributes is disabled by default. AIS properties and TM attributes are neither mapped nor propagated to the TM entries. To enable this feature, set the following TM property:

```
map_ais_property_to_same_named_tm_attribute=[true|false]:
```

Determines whether the AIS property values are used to populate the TM attribute values where the AIS properties have been mapped to TM attributes by name.

Mapping Segment Attributes to TM Entry Attributes

At times, it might be desirable to propagate segment-level attributes to the TM entries that are generated from them. For instance, the translator or reviewer may have provided useful notes in the comments section of the segment. These notes or comments might provide useful insight to explain why the source was translated a particular way. Translators leveraging this translation might find it useful to review these notes when deciding whether or not the translation is in fact appropriate for their purposes.

The only segment-level attribute that WorldServer currently allows to be auto-propagated to the TM entry is the comments attribute. Additionally, WorldServer allows the comment stored in the TM entry to be propagated back to the segment whenever the TM entry is leveraged as a 100% or SP/ICE match.

This feature is not enabled by default. In order to enable this feature, a specially named TM entry attribute must be constructed, which will hold this value. The attribute must be a text based attribute, and the internal (API) name must be `_tm_entry_comment`. Once this attribute has been created, by selecting **TM Entry** from the Attributes “Custom component type” Applicable Object Types drop-down list, WorldServer will automatically propagate comments from the segment to the TM entry generated during the TM update process. Similarly, the TM entry comment will be propagated to the segment during the leverage process if a 100% or better match is scored.

Using TM Entry Attributes

You can use the information in attributes in many ways. The information collected will be available to users and various WorldServer and custom processes. It can be used minimally for the following purposes:

- **Review and analysis** – The information can be viewed within WorldServer and, depending on the nature of the information, one might make a quality judgment on the entry. Additionally, the information could be used to generate reports.
- **Impacting leverage ranking** – Certain attributes, if mapped to AIS properties, can be used to influence the ranking of TM matches. See *TM Matching Based on Metadata* on page 93.

Chapter 13

TM Matching Based on Metadata

Topics:

This chapter explains how you can take advantage of the WorldServer metadata-based ranking feature that allows you to define TM attributes and AIS properties that represent user-defined logical boundaries.

Overview of Ranking Based on Metadata

Customers often migrate to WorldServer with the expectation of increasing leveragability by centralizing all their disparate TMs. However, this activity is not always without cost. While you can effectively leverage across all TMs, you lose the logical data boundaries that may have existed when TMs were originally generated for a specific purpose.

For instance, a company might have created separate TMs for each of the company’s products. While operating against a single product TM prevented leveraging translation work done across other products, it did ensure that all matches were in the context of that particular product. Merging multiple product TMs into a single TM no longer ensures this product context for the matches that are chosen. The end result is an increase in leveragability at the cost of inter-product translation quality.

For other companies, the boundary may have been established for a reason other than product separation. However, in each case, centralization may lead to a similar impact on quality.

WorldServer attempts to address this issue with its TM group technology, which allows multiple TMs to be grouped together and treated as a single TM entry. In keeping with the above example, TMs can be created for each given product. The set of TMs can then be grouped together to form a single logical TM. The first TM in the group contains the preferred translations. However, this means that you’d need to create a TM group for each product, so that in each product context, the specific product TM is the primary TM for that group. This solution is fine for a small set of TMs. However, the burden of management grows with each new product. Additionally, you may see performance degradation when a large number of TMs are in a single TM group.

As an alternative, you can take advantage of the WorldServer metadata-based ranking feature. This feature allows you to define TM attributes and AIS properties that represent user-defined logical boundaries. For instance, to align translatable content and its translations by “business group”, you can create an AIS property that defines the value for a collection of assets. You would also define a TM attribute that stores this value for each generated TM entry. After the mapped AIS properties and TM attributes are defined, the lookup process can compare the values associated with the asset to that of the TM entries being matched. The leverage process will prefer the TM entry that contains mapped attribute values that match the mapped AIS properties of the asset being leveraged.

As an example, suppose you are searching for the match of

```
Please specify the protocol to be used.
```

— which has no SID match currently defined, and which has `product=StarServer`, `version=4.3`, `file=string.properties`. Suppose there are two perfect text matches for

```
Please specify the protocol to be used.
```

— one with `product=Report Engine` and the other with `product=StarServer`. The match for `product=StarServer` is preferred.

There are two potential ways that this feature could have been addressed:

- related order of precedence

  The preferred value for a particular metadata attribute depends on the asset being processed. For instance, if the asset belongs to a particular product group, then the leverage process should prefer TM entries that are related explicitly to that product group.
- absolute order of precedence

WorldServer uses the *related-order-of-precedence* strategy.

Enabling or Disabling Ranking Based on Metadata

Metadata-based ranking is disabled by default. To use this feature, AIS properties need to be mapped to TM attributes, and the following TM configuration property must be explicitly enabled.

```
rank_based_on_ais_tm_attribute_match=[true|false]
```

This property determines whether the system should allow for mapped attributes to affect the ranking process.

Chapter 14

Segment Repair Technology

Topics:

This chapter describes how to configure WorldServer to *repair* differences between a source lookup segment and a source segment stored in the TM.

Where to Set Properties

Set all properties described in this section in the `tm.properties` file. The file resides in the same directory as other WorldServer properties files:

```
<WorldServer Installation Directory>\web-inf\classes\config
```

For more information about `tm.properties`, see *Configuring TM Properties* on page 144.

Overview of Segment Repairing in WorldServer

During the lookup process, if attempts to find ICE matches and exact matches fail, repairs are attempted to improve match results. Often, differences between a source lookup segment and a source segment stored in the TM can easily be repaired. In some cases, these differences can be automatically repaired, minimizing the work required of the translators.

Repairs are in-memory changes; they are never automatically stored in TM translation records. If the translator agrees to the repairs, the translator can choose to add new translation records to the TM when the asset being translated is saved.

The repair process may or may not be able to repair segments to 100% levels. If the initial unrepaired segment score was below the configured threshold, then WorldServer TM will never allow it to be scored up to 100%. Nonetheless, for segments where the initial differences are minor, the repair process may be able to repair many of the segments, increasing the ability to leverage the TM, and decreasing translation costs.

Master Option: Determines Whether to Prevent All Repairs

You can set a master option which determines whether to prevent all repairs. To do so use the following property:

```
prevent_all_segment_repairs = true or false
```

If you set the value to `true`, then all repairs are disabled, regardless of any individual settings you have configured. The only exception is the placeholder repair. However, disabling all repairs prevents the placeholder repair from affecting the final score.

The default value for this option is `false` which means that all repairs are allowed, and the system works with any individual settings you have configured.

If you do allow repairs, you can configure the following repair options for each source-to-target locale pairing:

- **number transform repair** – Attempts to repair numeric value differences
- **number deletion repair** – Attempts to remove non-required numerical data
- **placeholder repair** – Attempts to repair placeholder differences
- **word transform repair** – Attempts to repair “untranslated” content
- **word deletion repair** – Attempts to delete extra words
- **punctuation repair** – Attempts to correct boundary punctuation differences
- **capitalization repair** – Attempts to enforce consistency between replacement text and original text

Locale-to-Locale Mapping of Repairs

Many of the WorldServer repairs are most effective within a specific context, defined by the source and target languages. Often repairs are designed to capitalize on common elements shared between two or more languages or to exploit differences. If the two languages involved do not share the commonality implied by a specific repair, the application of the repair may do more harm than good. The customer needs to evaluate the usefulness of each given repair.

You can enable or disable each repair for specific source-to-target locale mappings.

Syntax

To map repairs to a specific source-to-target locale pairing, add the appropriate configuration options to the `tm.properties` file. The general syntax for configuring repair options is as follows:

```
{[source language id].}[repair property].[attribute name]={values}*
```

where:

- **source language id**

  2- or 3-letter symbol for the source language; for example, `en`, `fr`. If the source language is not specified, then the property is applied to all source languages that do not have a specific setting.
- **repair property**

  identifier for the type of repair to be configured.
- **attribute name**

  identifies the property attribute to be configured. This may refer to a repair specific property, or to a common repair property.
- **values**

  the set of appropriate values for the attribute being configured.
- The left (key) values are case sensitive. The right values are not.

All repairs support the following global attributes:

- `target_languages`

  Specifies which target languages to use with this repair for the given source language:

  - Ensure that the values are the standard 2- or 3-letter symbol representing the language. If you list multiple values, separate them with commas.
  - The value `all` denotes that the setting is for all languages.
  - The value "" (blank) denotes that no target languages are valid with the specified source language.
- `enabled`

  Specifies whether this repair should be enabled or disabled for the source language; values are `true` and `false`. (Any value other than `true` is interpreted as `false`.) You can disable this repair without changing this option by simply providing an empty set for the `target_language` property. However, for readability, we suggest that you explicitly set this value.

Example

Individual repairs may support additional repair-specific attributes. Default values are used only when the language-specific setting is omitted. Consider this set of sample entries:

```
punctuation.target_languages=
punctuation.enabled=false
en.punctuation.target_languages=DE,fr
en.punctuation.enabled=true
```

In this example, the punctuation repair is disabled for all languages other than English. For English, this repair can only be used when the target language is either German or French. The absence of a default configuration invokes the implicit default, which is to allow the full use of the repair, except where otherwise expressed.

Deprecated Properties

The following TM property entries are no longer supported by WorldServer.

- `allow_punctuation_repairs`
- `allow_number_repairs`

Core Supported Repairs

To effectively configure the repair options, it is important to understand them and their potential effect. The following sections provide details for each repair option.

Number Transform Repair (Enabled by Default)

Overview

Description	Attempts to repair numeric value differences.
Default	Enabled
Supported Attributes	`target_languages` `enabled`

Details

Number transformations occur when the lookup and the source contain different numbers. The repair process attempts to repair them. The underlying premise is that if the source and the target (translation) contain the same number, then both can be swapped for the number that appears in the lookup string. Consider the following simple example.

```
Lookup Segment: This has 1 number(s) different
Source Segment: This has 2 number(s) different
Target Segment: Thisa  haza 2 numbera different
Repaired Source Segment: This has 1 number(s) different
Repaired Target Segment:  Thisa  haza 1 numbera different
```

In mapping numbers between the lookup and the source, the order of the numbers is considered. However, when mapping numbers from the source to the target, only the instance of the number matters. In cases where there are multiple numbers, the repair process does not assume that the target (translation) text has the numbers ordered in the same way as the source text. Consider the following example.

```
Lookup Segment: 3 and 2
Source Segment: 2 and 3
Target Segment: 3 and 3
Repaired Source Segment: 2 and 2
Repaired Target Segment: 2 and 3
```

Important:

- This repair occurs only between locales that use the same numeric symbols for digits.
- Only western numbers are recognized.
- Number comparisons for swaps currently occur ignoring commas and decimals. For example, `99.5` matches `9.95` and to `9,95`. We will address this limitation in a future release.

Number Deletion Repair (Enabled by Default)

Overview

Description	Attempts to remove non-required numerical data.
Default	Enabled
Supported Attributes	`target_languages` `enabled`

Details

The number deletion repair attempts to remove numbers in the TM match that do not appear in the lookup segment.

In the following example, the numbers in the source and target match, and are therefore deleted.

```
Lookup Segment: There are more pies.
Source Segment: There are 2 more pies.
Target Segment: 2 pies are left.
Repaired Source Segment: There are more pies.
Repaired Target Segment:  pies are left.
```

Placeholder Repair (Enabled by Default)

Overview

Description	Attempts to repair placeholder differences.
Default	Enabled

Supported Attributes	`target_languages`
	`enabled`
	`unsafe_repair_penalty`

Details

WorldServer requires that the number of placeholders in the source and target text match for internal alignment. If the lookup has more placeholders than the source and targets, then additional placeholders are added to the source and target of the TM match during the repair process. If the lookup has fewer placeholders, then placeholders will be stripped from the source and target segments. Where possible, the repair process attempts to remove and place the placeholder indicators in the appropriate locations within the segments. When this not possible, the placeholder indicators may be placed at the beginning or end of the segment.

In the following example, the lookup segment contains zero placeholders, so to improve the match and facilitate internal alignment, the placeholders have been stripped out of the source and target segments:

```
Lookup Segment: This is simple.
Source Segment: This {1}is{2} simple.
Target Segment: Must this {1}be{2} so simple?
Repaired Source Segment: This is simple.
Repaired Target Segment: Must this be so simple?
```

In the following example, the lookup has additional placeholders. The repair process adds additional ones to both the source and target segments. Location is based on segment similarities and differences. This scenario tries to match up the outside placeholders based on the lookup segment.

```
Lookup Segment: {1}{2}This is foo.{3}{4}
Source Segment: {1}This is foo.{2}
Target Segment: {1}Foo is here.{2}
Repaired Source Segment: {1}{2}This is foo.{3}{4}
Repaired Target Segment: {1}{2}Foo is here.{3}{4}
```

In the following example, the lookup has additional placeholders. The repair process adds additional ones to both the source and target segments. Location is based on segment similarities and differences. The target repairs are done somewhat arbitrarily in this case since there is no way to exactly determine the location of the placeholders.

```
Lookup Segment: {1}This{2} is {3}foo{4}
Source Segment: {1}This is foo{2}
Target Segment: {1}Foo is this{2}
Repaired Source Segment: {1}This{2} is {3} foo{4}
Repaired Target Segment: {1}{2}{3}Foo is this{4}
```

In the following example, the placeholder indicators have been shifted to match the lookup.

```
Lookup Segment: {1}{2}This is foo.
Source Segment: {1}This is foo.{2}
Target Segment: {1}Foo is this.{2}
Repaired Source Segment: {1}{2}This is foo.
Repaired Target Segment: {1}{2}Foo is this.
```

Placeholder placement is based on the lookup segments because the lookup segment is the segment being translated. In the rendered document, the actual values substituted for the placeholders will come from the document or asset associated with the lookup segments.

Safe Placeholder Repair

The safe placeholder repair identifies a subset of placeholder repairs that are considered as safe (or minimal risk) placeholder repairs. These repairs involve placeholder scenarios that are highly deterministic in regards to the proper placement and removal of placeholders in the TM match being repaired. The current implementation restricts the designation of safe placeholder repairs to TM match segments that contain boundary placeholders only or to cases where all placeholders are to be removed. Cases involving the placement of embedded placeholders are deemed not safe.

In the following example, all of the placeholders are boundary placeholders, and so this is considered a safe repair:

```
Lookup Segment: {1}{2}This is foo.{3}{4}
Source Segment: {1}This is foo.{2}
Target Segment: {1}Foo is here.{2}
Repaired Source Segment: {1}{2}This is foo.{3}{4}
Repaired Target Segment: {1}{2}Foo is here.{3}{4}
```

The following example is not considered a safe repair. The proper placement of internal (embedded) placeholders within the target segment is hard to determine definitively.

```
Lookup Segment: {1}This{2} is {3}foo{4}
Source Segment: {1}This is foo{2}
Target Segment: {1}Foo is this{2}
Repaired Source Segment: {1}This{2} is {3} foo{4}
Repaired Target Segment: {1}{2}{3}Foo is this{4}
```

Unsafe repair penalty attribute

The `unsafe_repair_penalty` attribute (default value is 0.0) – allows the user to set a final scoring penalty to be assessed if the placeholder repair that has been applied is not deemed by the system as safe. Safe placeholder repair support is not an explicit repair option, although the system supports identifying these types of repair conditions. This option effectively allows the user to prevent unsafe repairs from being scored up to 100% matches. Set the value between 0.0 and 1.0. A practical value is 0.01, which translates into a 1% final score penalty.

Recommendation

SDL encourages customers to analyze the quality of all placeholder repairs. We also recommend that you at least accept safe placeholder repairs. In fact, in many cases, placeholder repairs that have not been explicitly deemed safe have in fact resulted in high quality results. Given the potential benefit of these repairs, we recommend that you evaluate the samples from your data scenarios. While the placeholder repair is required for WorldServer internal alignment, you can choose to completely disable this repair's impact on the final score or simply add a penalty for unsafe placeholder repairs.

Word Transform Repair (Disabled by Default)

Overview

Description	Attempts to repair "untranslated" content.
Default	Disabled
Supported Attributes	`target_languages` `enabled`

Details

The word transform repair seeks to identify content that should be translated as-is into the target segment. The current implementation is limited to identifying differences between the lookup segment and the TM match, and to determining whether the difference has been translated verbatim in the target segment. If so, it performs a substitution.

In the following example, the untranslated block is the text between `more` and `pies`.

```
Lookup Segment: There are more crazy pies.
Source Segment: There are more amazing pies.
Target Segment: Amazing pies are still there.
Repaired Source Segment: There are more crazy pies.
Repaired Target Segment: Crazy pies are still there.
```

The two words from the lookup segment were found in the source segment in the same order, but the text between the two words in the source was different. The repair process determines whether a repair is possible. The differing text is `amazing`, and since it exists in the target, the transformation to `crazy` (from the lookup) is done. This repair assumes that since the original text was untranslated in the target segment, the new replacement text does not have to be translated either. This may or may not be correct and it needs to be validated by the translator.

In the following example, the transform is not possible since the differing text in the source, `of the amazing`, is not found completely in the target. As a result, no transform is done.

```
Lookup Segment: There are more crazy pies.
Source Segment: There are more of the amazing pies.
Target Segment: Amazing pies are still there.
Repaired Source Segment: There are more of the amazing pies.
Repaired Target Segment: Amazing pies are still there.
```

Recommendation

Note that this initial implementation is simplistic and is not universally appropriate. Give careful thought to when to enable this repair. Do not use it when translating between languages that regularly share vocabulary.

Word Delete Repair (Enabled by Default)

Overview

Description	Attempts to delete extra words.
Default	Enabled

Supported Attributes	`target_languages` `enabled`

Details

The word delete repair process attempts to remove text that is determined to be extra. If the source contains additional text at the beginning of the segment, between two words that matched in the lookup segment, or at the end of the segment, then the repair process attempts to delete the words. Text segments are removed only if they exist as untranslated text, found both in the source and target segments.

In the following example, the extra word is untranslated text, and thus is deleted:

```
Lookup Segment: I am a sentence.
Source Segment: I am a longer sentence.
Target Segment: I happen to contain the word longer as well.
Repaired Source Segment: I am a sentence.
Repaired Target Segment: I happen to contain the word as well.
```

In the following example, the extra words are not untranslated text.

```
Lookup Segment: There are more pies.
Source Segment: There are more of the amazing pies.
Target Segment: Amazing pies are still there.
Repaired Source Segment: There are more of the amazing pies.
Repaired Target Segment: Amazing pies are still there.
```

Recommendation

This repair is deemed reasonably safe for general-purpose use. The current implementation will be applied to untranslated content only.

(Outer) Punctuation Repair (Enabled by Default)

Overview

Description	Attempts to correct boundary punctuation differences.
Default	Enabled
Supported Attributes	`target_languages` `enabled`

Details

WorldServer TM currently supports punctuation repairs, but they are very limited, and they are not appropriate for certain locale-to-locale translations. This repair assumes that the two locales involved contain the same punctuation symbols and that they are used in the same way. The current support for punctuation repair attempts to align the beginning and ending punctuation of the source and target of the TM match with that of the lookup segment.

In the following example, the beginning and ending punctuations of the lookup segment replace those from the source and target.

```
Lookup Segment: ‘This is boring!’
Source Segment: This game is boring.
Target Segment: This seems to be something.
Repaired Source Segment: ‘This game is boring!’
Repaired Target Segment: ‘This seems to be something!’
```

Recommendation

This repair is not appropriate for all source-to-target locale combinations. However, the implementation provides an additional safeguard, which allows only the outer punctuations to be repaired in the cases where the outer punctuations match for the source and target text of the TM match.

Capitalization Repair (Enabled by Default)

Overview

Description	Attempts to enforce consistency between replacement text and original text.
Default	Enabled
Supported Attributes	`target_languages` `enabled`

Details

The repair process does not currently repair capitalization to match the lookup segment. During word transformations, it attempts to maintain the capitalization of the original words from the source and target. If the word being replaced is capitalized, the new word will be capitalized. If the entire text is all capitalized, the replacement text will be all capitalized. If the original text is mixed, only the first letter of the new text will be capitalized (but only if the first letter of the original text was capitalized).

Consider the following examples:

```
Lookup Segment: There are more crazy pies.
Source Segment: There are more amazing pies.
Target Segment: Amazing pies are still there.
Repaired Source Segment: There are more crazy pies.
Repaired Target Segment: Crazy pies are still there.
```

```
Lookup Segment: There are more crazy pies.
Source Segment: There are more aMazing pies.
Target Segment: AMAZING pies are still there.
Repaired Source Segment: “There are more crazy pies.
Repaired Target Segment: “CRAZY pies are still there.
```

```
Lookup Segment: There are more crazy strawberry pies.
Source Segment: There are more AMAZING APPLE pies.
Target Segment: Amazing Apple pies are still there.
Repaired Source Segment: There are more CRAZY STRAWBERRY pies.
Repaired Target Segment: Crazy strawberry pies are still there.
```

Note:

- If all the characters are not capitalized in all words in the text block, then, at best, only the first word after the transformation will be capitalized.
- The current implementation of this repair is tightly coupled to the word transform repair. This is somewhat highlighted in the above examples, which all could be used as word transform examples. Disabling the word transform repair also disables this repair. It attempts to ensure capitalization consistency with the text in the segment being replaced.

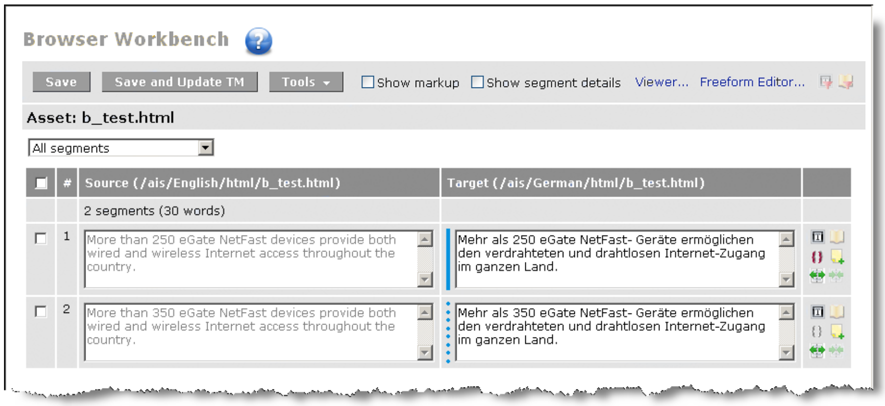
Recommendation

SDL encourages you to use this repair whenever the word transform repair is used. This repair to minimizes the need to make additional edits related to capitalization. However, there may be cases where you do not wish to apply the heuristic, in which case you should disable this repair.

Highlighting Repaired 100% Matches

Within WorldServer, all 100% matches are displayed with a blue status bar. A match can be at 100% if it is an exact match or if it was a high fuzzy match that was elevated to a 100% match as the result of automatic repairs. In earlier versions, there was no way to differentiate between the two types of matches. A recent enhancement allows the user to visually identify which 100% matches have been repaired.

Segments with repaired 100% matches are now illustrated with a dotted blue line, as illustrated in the following image. The solid blue line is reserved for natural 100% matches.

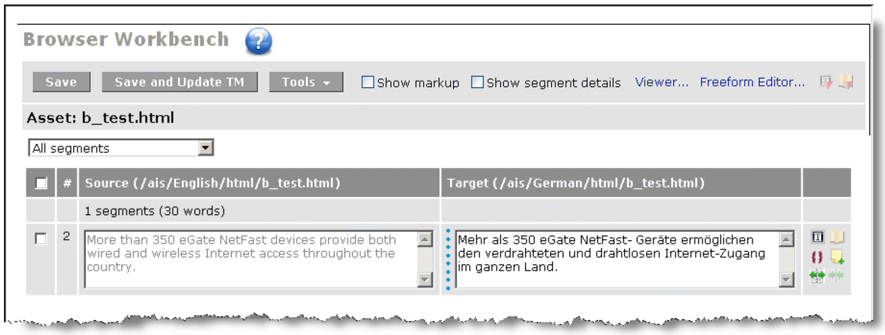


Asset: b_test.html

	#	Source (/ais/English/html/b_test.html)	Target (/ais/German/html/b_test.html)
		2 segments (30 words)	
	1	More than 250 eGate NetFast devices provide both wired and wireless Internet access throughout the country.	Mehr als 250 eGate NetFast- Geräte ermöglichen den verdrahteten und drahtlosen Internet-Zugang im ganzen Land.
	2	More than 350 eGate NetFast devices provide both wired and wireless Internet access throughout the country.	Mehr als 350 eGate NetFast- Geräte ermöglichen den verdrahteten und drahtlosen Internet-Zugang im ganzen Land.

Figure 21: Browser Workbench Segmentation Example 7

A filter option allows the user to view only those segments that have been populated with repaired 100% matches.



Asset: b_test.html

	#	Source (/ais/English/html/b_test.html)	Target (/ais/German/html/b_test.html)
		1 segments (30 words)	
	2	More than 350 eGate NetFast devices provide both wired and wireless Internet access throughout the country.	Mehr als 350 eGate NetFast- Geräte ermöglichen den verdrahteten und drahtlosen Internet-Zugang im ganzen Land.

Figure 22: Browser Workbench Segmentation Example 8

Custom Repairs

In addition to the core supported repair algorithms, WorldServer supports the creation of custom repair algorithms that can be uploaded and used in tandem to the core implementations. The new uploadable component is supported via the WorldServer SDK support. For detailed information on the creation and deployment of custom repair implementations, please refer to the *WorldServer SDK User Guide*.

The ability to create custom repairs is a very powerful feature. It facilitates the ability to automatic handle translation corrections that WorldServer currently does not support. For instance, operations like number conversions and value localizations can be facilitated via this framework.

Chapter

15

Match Scoring Configuration

Topics:

This chapter describes how to configure Translation Memory (TM) to affect scoring during fuzzy matching.

Overview of TM Fuzzy Match Scoring

The scoring process for fuzzy matching supports the following match penalties:

- configurable penalties for punctuation and white spaces
- configurable penalties for capitalization
- configurable penalties for non-matching placeholders

These settings allow the implementation to generate scores that are more representative of the actual comparative effort involved in transforming one segment to another. To improve overall scoring, the weights used to compute the final scores are available for customization. The current implementation provides support for variable weighting of segment elements. The WorldServer Translation Memory (TM) recognizes these segment elements:

- codes or placeholders
- words
- numbers

In the current implementation, these elements can have different weights. For example, you may want a placeholder code and a word to have different weights. Similarly, translating a number versus a word may not represent the same effort. As a result, you may want to assign a smaller or larger weight to numbers. The implementation also provides support for differentiating between inner and outer placeholders.

The remainder of this chapter describes how to perform these changes and the impact of the changes on scoring. This chapter concludes with a table illustrating how the new scoring algorithm applies to different segments.

How TM Match Scoring Works

To understand how WorldServer scores matches, you need to understand its concepts of weights and penalties.

TM Weighted Scoring

The scoring algorithm calculates a percentage effort score based on the cheapest path of transforming one segment into another. The underlying process uses the string difference algorithm.

The basic idea is that the system associates a cost to each step required to transform one segment to the other based on the elements that make up the segment. The cost is a factor of the transformation required. For example:

- Inserting a word may have an associated cost of 1 unit of work.
- Correcting the punctuation following a word may have a significantly smaller cost.
- Matching elements yield a zero-cost edit.

The cost of each edit is averaged out over the total number of edits to yield a final score that represents the transformation effort.

WorldServer translation memory currently supports the transforms described in the following table. There is no plan to make changes to this table.

Transform	Default Cost	Configurable	Notes
delete element	element weight	No	Element weights are configurable, but a delete always costs the weight of the element.
insert element	element weight	No	(same as for delete elment)

Transform	Default Cost	Configurable	Notes
keep element	no cost	No	The elements match. The match is determined independently of surrounding (preceding or following) punctuation or whitespace.

The following table describes the supported correction penalties:

Correction Penalty	Default Cost	Configurable	Notes
Capitalization	.01 unit	Yes	Applies only to words.
Punctuation	.005 unit	Yes	Covers all non-alphanumeric data, not just punctuation. Penalty is exacted whenever a prefix or suffix differs between the compared elements.
Placeholder	outer: .01 unit inner: .25 unit	Yes	Represents a minor penalty assessed when the two placeholders being compared have different sequence numbers.

The maximum cost of an edit step is the weight of the element being transformed. Insert and delete transforms automatically result in a maximum cost. As a result, penalties apply only to keep transforms.

The penalties are weighted, not absolute. These penalties are applied to the transform step, and thus are averaged across all transform steps.

Configurable Weights

One major difference in the updated implementation is in the level of control now supported by the scoring process. The segments are decomposed into elements, where each element type is assigned a weight that represents its maximum effort. (In the old implementation, all elements were equally weighted.) The new implementation provides granular control over the weight on elements within the segment.

The following segment elements are recognized and the weights for most of them can be configured individually:

- **word**

  Represents one unit of work, and is constant; all other element weights are assigned relative to a word unit.

- **number**

  By default, numbers have less weight than words, but this setting can be changed. We expect that a difference in numbers is not as important as a difference between words.

- **outer placeholder**

  The implementation allows for leading and trailing placeholders to be weighted differently from words, as well as from inner placeholders.

- **inner placeholders**

  Placeholders where at least one non-placeholder element (such as a word or number) occurs before and after at some point in the segment. For example, consider the text `Please click <a href=”…” >this text </a> to advance forward.` The anchor tags are converted into placeholders, which are both considered inner tags.

The following table shows the configurable segment element weights along with their default values. You can change these values by editing the `tm.properties` file.

TM Property	Value Set (Default)	Description
`tm_score_number_weight`	0 – 1 (.2)	The weight assigned to a numeric element. By default, number differences are weighted less than word differences.
`tm_score_outer_placeholder_weight`	0 – 1 (.1)	Outside placeholders by default are weighted less than inner placeholders, and both are weighted significantly less then words and numbers.
`tm_score_inner_placeholder_weight`	0 – 1 (.25)	The weight assigned to an inner placeholder. By default, inner placeholders require more review than outer placeholders.

Configurable Penalties

The following table identifies the scoring penalties supported by WorldServer. Penalties are assessed against each segment element based on the specific differences between the lookup segment and the TM match segment. You can override the default penalty values by editing the `tm.properties` file.

TM Property	Value Set (Default)	Description
`tm_score_capitalization_penalty`	0 – 1 (.01)	This penalty is assessed when the word elements being compared are identical except for capitalization differences. By default, the penalty is less than the full word difference cost.
`tm_score_punctuation_penalty`	0 – 1 (.005)	This penalty is assessed when the elements being compared are identical except for surrounding punctuation differences. By default, the penalty is less than the full word difference cost. Note that this penalty may be assessed twice for a given set of compared elements since the leading and following punctuations are handled separately.
`tm_score_placeholder_sequence_penalty`	0 - 1 (.05)	This penalty is assessed when the elements being compared are both placeholder values, but they have different index values. (Having different index values means that one of the segments has more placeholders than the other up to the point being considered.)

Values of zero disable the penalties. Penalties reach a maximum at the weight of the element being penalized.

Placeholder Repair Note

The repair process currently auto-corrects placeholder issues regardless of whether repairs are enabled. This is necessary for placeholder matching within the TM to function correctly. As a result, the TM segment that is displayed as a hit always reflects the placeholder repairs. However, the final score does not reflect the repairs if repairs have been disabled. In this case, if the placeholders yield the only differences in the lookup and hit segments, then it might not be clear why the match is not 100% until the user looks at the original text for the hit segment.

Sample Cases

The following examples show how the configurable options can affect the scoring results.

Option	**Scoring Scenario 1**	**Scoring Scenario 2**
Weights	• word = 1 unit • number = 1 unit • outer placeholder = .25 • inner placeholder = .5	• word = 1 unit • number = 1 unit • outer placeholder = .05 • inner placeholder = .025
Penalties	• capitalization = .2 • punctuation (non-alphanumeric) = .1 • non-matching placeholder sequence = .1	• capitalization = .05 • punctuation (non-alphanumeric) = .01 • non-matching placeholder sequence = .05
Summary	This scenario has a big penalty for capitalization. For segments with few words, this can have a big impact on the final score. For extremely long segments, where few words are penalized, the effect is smaller.	This scenario registers only about a 5% penalty per word.
Result	If the only issues in the two segments being compared are capitalization, the worst possible score is an 80.0% match. (Capitalization exacts a .2 or 20% penalty)	If the only issues in the two segments being compared is capitalization, the worst possible score is a 95.0% match.

The results in the following table reflect the scenarios described in the previous table. Also, the scores represent no repairs being done.

Lookup Segment	**Hit Segment**	**Scenario 1 Results**	**Scenario 2 Results**
"this is a segment"	"this is a segment"	100	100
"this is a segment"	"This Is A Segment"	80	95
"this is a segment"	"{1}this is a segment"	94.1	98.7
"this is a segment"	"this is a{3} segment"	88.9	99.3
"this is a segment"	"this is a segment."	97.5	99.7
"this is a segment"	"this 'is' a segment"	90	99
"this is a segment"	"this segment"	50	50
"this is a segment"	"Need more Calgon!"	0	0
"this is a segment"	"a segment merged together"	33.3	33.3
"this is a segment"	"this segment is a"	60	60
"{1}this is a{2} segment"	"this is a segment"	84.2	98.1
"{1}this is a{2} segment"	"This Is A Segment"	67.3	93.2
"{1}this is a{2} segment"	"{1}this is a segment"	89.4	99.3

Lookup Segment	**Hit Segment**	**Scenario 1 Results**	**Scenario 2 Results**
"{1}this is a{2} segment"	"this is a{3} segment"	92.6	98.1
"{1}this is a{2} segment"	"this is a segment."	82.1	97.9
"{1}this is a{2} segment"	"this 'is' a segment"	75.7	97.1
"{1}this is a{2} segment"	"this segment"	42.1	49.0
"{1}this is a{2} segment"	"Need more Calgon!"	0	0
"{1}this is a{2} segment"	"a segment merged together"	29.6	32.9
"{1}this is a{2} segment"	"this segment is a"	52.1	59.1

Note that the drastic difference in scoring for Scenarios 1 and 2 effectively illustrate the control available through the configurable options. Companies with varying average size segments can align weights and penalties to emphasize specific differences between the lookup and hit segments.

Example of How Word Differences Affect Match Score

The following table explains how the match score is computed when the lookup and hit segments differ only in the number of words or if some words are different.

Lookup Segment	**Hit Segment**	**Difference**	**Score**	**Score derivation explanation**
"this is a segment"	"this is also a segment"	Hit segment has an additional word.	80%	Lookup segment has 4 out of five matching words. (`4/5 = 80%`).
"this is also a segment"	"this is a segment"	Lookup segment has an additional word.	80%	Hit segment has 5 words, Lookup 4 words. (`4/5 = 80%`)
"this is a segment"	"this is a sentence"	Segments have one different word.	75%	3 of 4 words match. (`3/4 = 75%`)

Note that the scoring is order-dependent. Two segments might have the same words, but if they are in a different order, they will not have a `100%` match score. For example, "`John saw Jim`" and "`Jim saw John`" would not be a have a `100%` match score.

Example of How Capitalization Differences Affect Match Score

The following table explains how the match score is computed when the lookup and hit segments differ only in that some words are capitalized in one and not the other. Each table represents a different capitalization penalty.

`tm_score_capitalization_penalty=0.01` (the default)

Lookup Segment	**Hit Segment**	**Difference**	**Score**	**Score derivation explanation**
"This is a segment"	"This is a Segment"	Hit segment has one word capitalized that Lookup didn't.	99.75%	The one capitalized word difference is penalized 1%, and that word is 25% (one of four words) of the score. So, 3 of 4 match (totaling 75%) and the

Lookup Segment	Hit Segment	Difference	Score	Score derivation explanation
				other 25% is penalized 1% (so that word would be `25 - .25 = 24.75%`). So the total score would be `75% + 24.75% = 99.75%`.
"this is a segment"	"This Is A Segment"	Hit segment has four capitalized words, the Lookup segment none..	99%	This penalty will be for each of the 4 words, so the score will be `4 x 24.75 = 99%`.

`tm_score_capitalization_penalty=0.2`

Lookup Segment	Hit Segment	Difference	Score	Score derivation explanation
"This is a segment"	"This is a Segment"	Hit segment has one word capitalized that Lookup didn't.	95%	The one capitalized word difference is penalized 20%, and that word is 25% (one of four words) of the score. `20% x 25% = 5%`. So the score is `100% - 5% = 95%`.
"this is a segment"	"This Is A Segment"	Hit segment has four capitalized words, the Lookup segment none..	80%	This penalty will be 4 times the previous one, so it would be 20%; so the score would be `100% – 20% = 80%`.

Example of How Placeholder Weights Affect Match Score

The following table explains how the inner placeholder weight in the examples above are computed.

`tm_score_inner_placeholder_weight=0.5`

Lookup Segment	Hit Segment	Difference	Score	Score derivation explanation
"this is a segment"	"this is a{3} segment"	Hit segment has an inner placeholder.	88.9%	Inner placeholder is weighed at `.5` of a word, so match is `4` out of `4.5`. `4/4.5 = 0.8888 = 88.9%`.

Scope of Settings

In the implementation described in this document, the properties are assigned at a system-wide level, and apply to all users across all projects. While setting default values for the system is a good idea in that it will ensure consistent scoring across the enterprise, providing overrides at the project level might be worth considering. Potential project-level support is outside the scope of this document.

Chapter 16

Segment Identifier (SID) Support

Topics:

Segment identifiers (SIDs) allow customers to explicitly define the context for a given segment. WorldServer has added support that allows these values to be used, provided that the proper filter requirements have been met. This chapter provides more extensive details on the background and use of SIDs within the WorldServer environment.

Definitions

This section describes what SIDs are and the different types.

What are Segment Identifiers (SIDs) and how are they used?

- SIDs, in general, define the context for translatable content.
- SIDs are marker tags that define segment boundaries. The tags are placed around source content within source documents. Content between an opening and a closing SID marker is considered a single segment.
- SIDs help identify translation pairs. During the leverage process, SIDs are used to locate the best TM match for the SID segment.
- SIDs help define translation pairs. During the TM update process, SIDs are stored with the associated TM entry, thus allowing it to be used in subsequent SID-based leverage processes.

Types of SIDs

There are essentially two types of SIDs.

- Non-Unique SIDs (NUSIDs)

  An SID that is not necessarily unique, and can be associated with multiple translation pairs. It defines the context for translating the given content only. In such cases, the combination of the SID and the segment source text collectively resolve to a single translation pair. This is the less restrictive type of SID. No unique constraint needs to be enforced on the SID column. Regarding TM matches, the SID alone does not resolve into a single TM match. The SID (which defines context) needs to be combined with text to get a single TM entry. WorldServer supports this type of SID.
- Unique SIDs (USIDs)

  An SID that is globally unique, and thus can be associated with a single translation pair. This is the more restrictive type. A unique constraint needs to be associated with the SID column. In this case, SIDs define both the context and content. The idea is that a single SID should resolve into a single TM match. WorldServer does not support this type of SID.

Note: WorldServer supports only the NUSID configuration. (SIDs can be unique if the customer creates them to be unique, however, their uniqueness is neither required nor enforced by WorldServer.)

Overview of WorldServer Support of Segment Identifier (SID)

WorldServer supports SIDs in the following ways:

- Defines how SID markers should be used in customer content
- Leverages only properly defined SID markers
- Enforces boundaries defined by proper SID usage
- Associates defined SIDs with translation pairs
- Enforces supported uniqueness requirements
- Leverages SID content against SID TM entries

Some documents may not lend themselves to supporting SIDs. The use of SIDs within customer content should not be immediately visible to the end user. Document types that allow editors to embed non-viewable tags are ideal. For instance, XML and HTML documents are prime candidates. Other document types may or may not support embedding of information that could be used to define SID boundaries and levels.

SID support is not handled completely by WorldServer. WorldServer does not support the following features:

- Completely define how SIDs are represented in customer content
- Embed SID markers into customer content
- Enforce proper use of SID markers within content

Segment Preferred In-context Exact (SPICE) Match

The segment preferred in-context exact (SPICE) match is similar to the basic ICE mechanism. The SPICE match process supports another leverage level that is superior to the basic exact match supported by traditional TM systems. Instead of inferring context information from the surrounding segments and the containing document, the SPICE mechanism derives context explicitly from segment identifier (SID) tags embedded within the document. The SID defines the intended context, and thus the combination of the SID and the source segment text correlates to a specific translation within the TM regardless of the document within which it is embedded.

SPICE matches are global within the defined TM. There can be only one SPICE match candidate for any given segment/SID combination. This is different from the ICE mechanism, which allows for different documents to potentially have different translations for a common block of content. SPICE match usage is optional. System parameters can be configured to indicate whether SPICE matches are preferred over ICE matches.

SID-based Sorting/Filtering

The implementation of the SID as a standard attribute allows WorldServer to offer attribute options for SID values. Users can sort and filter TM records by SID values, and view SID values on all pages that currently provide this functionality for TM attributes. These include the following pages and processes:

- TM Tool Query Page
- TM Entries Page (Display Options)
- TM Filter Process

SID Configuration Options

You have some control over how segment matching is conducted.

SID vs. ICE Priority Settings

WorldServer allows the customer to control the order in which context-based matches are retrieved.

- disable/enable SID-based matching
- prefer ICE over SID-based matching
- prefer SID over ICE-based matching

Preferring the standard ICE matches means that you prefer the more localized usage context of translations. In this strategy, you might have SID-based matches that provide a default SPICE translation in the absence of a localized ICE match. However, as soon as there is a localized ICE match, it will be preferred.

Preferring SPICE matches over the standard ICE means that you want to establish more consistent, universal translations for content. The SID defines the context entirely, negating the normal match ranking strategy. Since there can only be one SPICE match for a given SID + Source text combination, there is no need for ranking. ICE matches become secondary, and are used only when SPICE matches are not available.

The preference options for ICE and SPICE matches may not result in significant leverage results in environments that almost entirely use SID-based content or in environments that use mainly non-SID-based content. For asset segments without associated SID values, WorldServer stores a copy of the translation for each asset.[12] Currently WorldServer does not store asset-based entries for segments containing SIDs when the translation has not been updated.

TM configuration options

TM configuration options are defined in `tm.properties`, using the following entry:

- `prefer_sid_over_ice_matches=true`

  When SPICE matching is enabled, prefer it over ICE by default. If `true`, look for SPICE match before looking for ICE match. If `false`, seek ICE match first.

Key Points:

SID support requires a custom filter, which does the following:

- Read the begin tag and end tag for a unique identifier from the source files.
- Activate SID in WorldServer with the unique identifier and associate the unique identifier with the TM segments.

The nature of the SID depends on the end requirements. Some customers choose SIDs that are unique for each segment. This is not a WorldServer requirement. Some customers reuse SID values across different source text.

The most important thing to determine is what format you will use for the SID. You can then create a utility that generates SIDs consistently and uniquely per segment.

[12] Storing a copy of each segment translation for each asset allows for identical content to be translated differently in different assets. This allows for historical assets to be retranslated in the same way as they were originally translated, even when the same content was more recently translated differently in another asset.

Chapter

17

Understanding Repetitions

Topics:

Some content may be reused within an asset and across a collection of assets. The translation of duplicated content should not result in the same translation cost as for the original content. Arguably, the content should not incur a translation cost at all, provided the translation of the original instance is readily available and appropriate for the new context of translation. However, the cost issue is beyond the scope of translation memory technology. Instead, the translation memory technology enables such cost discussions to be had by supporting and exposing a concept of repetitions.

The repetition calculation process identifies and counts repetitions across segments that cannot be fully leveraged by the TM. This means that both ICE and 100% matches are not included in the repetition counts. The idea is that ICE matches do not need to be reviewed at all, and represent no translation cost. 100% matches often do not incur a translation cost, depending on the vendor. At most, the customer may decide to have these 100% (non-ICE) matches reviewed, and the customer would be charged some minimal amount.

The motivation behind repetition counting is to reduce translation cost for the customer by preventing them from being charged full price for identical segments that must be translated. The first occurrence of a duplicated segment is considered the original or repeated segment, and it is scoped as normal. This means that the word count for the repeated segment will be attributed to the fuzzy match bucket representing the best TM fuzzy match for the segment, and thus will incur a translation cost relative to the translation effort. Subsequent occurrences of the segment are referred to as repetition segments. The word count for repetition segments are placed in the scoping bucket for repetitions. For example, if the segment "Oh what a beautiful morning" containing five words was repeated five times in an asset or set of assets, then the first occurrence would be scoped normally, and the additional occurrences (collectively containing 20 words) would be placed in the repetition bucket provided that they cannot be fully leveraged by the TM.

Note: It would not be helpful to include ICE and 100% matches in the repetition counts, because you would incur charges where there should be none (again assuming that repetitions are not free.) The implementation is geared toward reducing translation costs for words that are not fully leveraged by the TM.

Calculating Repetitions

WorldServer can identify all the duplicated segments (excluding all translated segments) within an asset and across the project. The following points describe the process:

- Segmentation and leverage occurs for each asset. The leverage process determines whether there are 100% or ICE matches available for each segment. If so, they will be applied and these segments will be attributed with a 100% score (and an ICE status if appropriate). If there are no ICE or 100% matches, then the best fuzzy score for each segment is recorded (0–99%) for scoping purposes.
- The repetition calculation process must be triggered. This must happen after segmentation and leveraging, and does not happen by default. The repetition counter goes through each asset and looks for duplicates among all segments that do not have a 100% score (which is why it is stated that only fuzzy matched segments are considered for duplication.) The first occurrence is designated as the *repeated segment*. Occurrences after the repeated segment are considered the *repetition segments*. The text used to check for duplications comes from the asset segment, not the fuzzy matches in the TM.

For repetitions to be calculated, the repetition process must be triggered, using one of these methods:

- Following the `Segment` automatic action, use the `Calculate Repetition` automatic action in a workflow. This will require all project tasks to be included in the repetition calculation process.
- Use SDK customizations to perform repetition calculations. (See the SDK documentation of more information.)

When calculating repetitions within a project, the repetition information persists for all assets involved. Recalculating repetitions will result in adding to any existing stored repetition data. Before recalculating repetitions for an asset or set of assets, the previous repetition results must be flushed. Currently, this requires forcing the assets to be resegmented (such as using the `Clear Cache` automatic action). The same applies to calculating repetitions via the SDK; however, the SDK provides explicit control of whether the repetition data is saved or discarded.

Evaluating Repetitions

The duplicate segment checking process in WorldServer is mostly straightforward. If one segment is an exact match of another segment, it is considered a repetition segment. For instance, the following segments are considered duplicates of each other:

```
{34}Progress Report{35}
{1}Progress Report{2}
```

The segments have exactly the same type, discarding the difference in the placeholder index. More correctly, the normalized forms of these segments are identical. The normalized form compares them after the placeholder index is converted. [13]

The following segments may optionally be considered as duplicates:

```
{34}Progress Report{35}
Progress Report{2}
{1}Progress Report
{34}{35}Progress Report{36}
Progress Report{2}
```

[13] The order of the placeholders would be important, but the order never comes into play for source assets because they are always numbered in increasing order. Only the target segments may contain placeholders in a reordered manner, but target segments are not processed for repetitions.

Outside placeholder differences are simple repair candidates that will be preformed by WorldServer reliably and automatically. For this reason, customers should consider reducing the outside placeholder penalty to zero. (See *Safe Placeholder Repair* on page 102 for more information.) If this is done, the above segments will be considered as duplicates.

Chapter 18

Word Breaking

Topics:

Word breaking attempts to identify word boundaries within textual content and presents the information to WorldServer for further processing. Word breaking is critical to translation memory (TM) and terminology database (TD) related searches (excluding ICE and pure exact match TM searches.) It defines the word units that are then used when creating shingles[14] for match searches.

Before WorldServer 8.0, WorldServer supported a single word breaker implementation that was used ubiquitously across all supported language. The process analyzed alphanumeric characters as well as whitespace and punctuation in order to determine where a word began, continued, and ended. The current implementation was designed with a bias toward English. However, this implementation was not sophisticated enough to handle all languages effectively. In particular, the core word breaker implementation did not handle character-based languages, such as Japanese, where a character represents a word. Additionally, the current word-breaking strategy may not be equally effective for all Western or word-based languages.

The core implementation is generally sufficient for WorldServer customers who use English as the source language for translation work. There are scenarios where this implementation breaks down. Attempts to perform *ad hoc* searches against certain target languages may break down, especially for character-based languages, where each character should represent a word. However, the driving statistics that drive a customer's ROI is based on the source language. As long as the source language is English, the impact of the English-centric word breaker is minimized.

However, the issue becomes more critical for customers who use a source language other than English. For instance, using Japanese as the source language dramatically exposed limitations of the core implementation. For Japanese and other character-based languages, the implementation almost completely eliminates fuzzy matches. Fortunately, the core implementation has been updated to identify and treat Chinese, Japanese and Korean characters each as separate words. This improves the fuzzy matching support for these languages. Additionally, WorldServer has introduced a framework that allows for a new word breaking implementation to be uploaded and used within WorldServer.

The following sections discuss WorldServer support for word breaking.

[14] *Shingles* refers to word runs or a series of words taken from a reference text.

WorldServer Standard Word Breaking

Character Classes

WorldServer introduces an internal character classification used for word breaking. Each character is placed into one of the following categories, based on character properties.

- Letter Characters (L)

  A character is considered to be a letter if and only if it is specified as a letter by the Unicode 2.0 standard (category `Lu`, `Ll`, `Lt`, `Lm`, or `Lo` in the Unicode specification data file). WorldServer uses the Java `Character.isLetter()` implementation to determine if a character is a letter.
- Digit Character (D)

  A character is considered to be a digit if it is not in the range `\u2000' <= ch <= '\u2FFF` and its Unicode name contains the word `DIGIT`. WorldServer uses the Java `Character.isDigit()` implementation to determine if a character is a digit.
- Middle-Word Characters (W)

  A character is considered to be a middle-word character if it is not a letter but it should be treated as one when it is encountered in the middle of a word. The set of middle-word characters is configurable in WorldServer. The default configuration includes common characters such as dash (`-`), apostrophe (`‘`) and “at” (`@`).
- Middle-Number Characters (N)

  A character is considered to be a middle-number character if it is not a digit but it should be treated as one when it is encountered in the middle of a number. The set of middle-number characters is configurable in WorldServer. The default configuration includes common characters such as dot (`.`) and comma (`,`).

During the word-breaking process, WorldServer scans an array of characters and determines word boundaries based on character classes. This information is then used to define basic elements used for word counting and segment comparisons. The basic elements that are identified are words, numbers, and placeholders. The additional information around and between these elements is categorized as punctuation (which includes whitespace).

Word and Number Definitions

A WorldServer word is defined as an array of letter (L) and middle-word (W) characters starting and ending with a letter. Using regular expression syntax, a word is defined as: `L[LW]*L`.

A WorldServer number is defined as an array of digits (D) and middle-number (N) characters starting and ending with a digit. Using regular expression syntax, a number is defined as: `D[DN]*D`.

The following table illustrates WorldServer word breaking algorithm.

I have a hat.	4 words
I’ve learned word-breaking.	3 words
He is 5-6 feet tall.	4 words and 2 numbers
Please send $10.27 and ¥10,000 to foo@bar.com.	5 words and 2 numbers

Word Counting

Word counting, typically used for scoping analysis, works hand-in-hand with word breaking. The following algorithm describes how the segment elements (words, numbers and placeholders) are used to calculate the word count for segments.

Word Counting Algorithm

WorldServer uses a simple and efficient word counting scheme. During the scoping process WorldServer breaks an asset into segments and then runs each segment through the word breaking process described above. After a list of word and number elements is generated, those words and numbers are counted using the following rules:

- Each word is counted as one point to be added to the total scoped value.
- For CJK (Chinese/Japanese/Korean), WorldServer has a special way to count words. Each character is considered a word. For these languages we are, effectively, counting characters. When a user sees “Words” in the WorldServer UI (for example, in scoping), for CJK source languages it actually means “Characters”. If a content is a mixture of CJK- and Latin-based languages, the appropriate word counting scheme is used for each language. For example, “WorldServer ” is counted as 6 words.
- Numbers are ignored unless a segment has no words.
- If a segment has no words but has at least one number the whole segment is counted as one word count.
- Wordless and numberless segments do not contribute to the scoped result.

This number counting scheme was chosen to adequately reflect a lower cost of number translation. In most cases numbers do not require any translation. Counting numbers in numerically oriented content numbers might increase the translation cost disproportional to the actual translation effort.

Customizing the Scoping Algorithm

The standard word breaking algorithm can be customized using an external properties file. Users can change the definition for middle word and number characters to adjust word breaking.

The scoping scheme can be customized by implementing a custom filter component that replaces the word counting algorithm. By implementing a custom filter and overwriting the default counting scheme a customer can implement any required counting scheme.

Custom Word Breaker Support

WorldServer now supports the creation and use of custom word breaker implementations via the WorldServer SDK. Customers can implement language-specific word breaking implementations to use in place of the core implementation. This provides a framework through which specialized word breaking algorithms and implementations can be used within WorldServer.

Evaluate the core word breaker implementation for your needs before you invest in a custom implementation. The simplest way to do this is to use the translation memory search tool in standard mode to perform some word-based lookup. Look for words that you know are in your translation memory. You might start by running a query for “ `*` ” to return the first 1000 records (or whatever maximum you set). If you are able to search for specific words and get back the expected results, then the word breaker is most likely sufficient for you in that language.

You can evaluate for any language you like; however, in general, you should focus on languages that will be used as the source language. As mentioned earlier, the core implementation does not work well for character-based languages such as Japanese and Chinese. This will affect your ability to do word-based searches on these languages. However, if these languages are not being used as source languages, there may be limited return from investing in custom word breaker implementations for these languages. Translation costs are generally driven from the scoping reports generated from source content.

See the WorldServer SDK documentation for details on creating and deploying custom word breaker implementations.

TRADOS vs. WorldServer Scoping

Word Counting Scheme

TRADOS uses a more simplistic word counting mechanism than WorldServer. It does not have a middle word character concept and counts contractions and hyphenated words as separate words. TRADOS also has a different number counting scheme. It counts numbers the same way as words if a segment has at least one word. Numbers in number-only segments are not counted. These differences cause a higher word count by TRADOS in comparison to WorldServer.

At the same time, TRADOS does not break words on the '_' character and considers strings separated by '_' as a single word. TRADOS is not consistent in counting numbers separated by markup. In some cases the numbers are counted and in some cases they are not. These changes might result lower word count for TRADOS in comparison to WorldServer.

These differences generally result in different word counts between WorldServer and TRADOS, as illustrated in the following table:

	WorldServer count	**TRADOS count**
I've learned word-breaking.	3	5
He is 5-6 feet tall.	4	5
The password is I_AM_BAD.	6	4
<b>Number of items:</b> 24.	3	3
<b>Number of items:</b> 24 (approx).	4	5

Repetition Counting Scheme

A segment is considered to be a repetition if the same segment has already occurred during the scoping process in either the same asset or in another asset. TRADOS compares segments exactly. A segment becomes a repetition only if it matches completely including number and placement of placeholders (but the contents of the placeholder does not have to match).

In WorldServer a segment is considered a repetition if a previously translated match can be used to generate a 100% match for this segment. Even if a segment does not contain the same text as a previously translated match it can be counted as a repetition. For example, if segments differ by leading and trailing placeholders, only the second segment is counted as a repetition because placeholder difference will be automatically repaired up to 100% match.

For example, consider two segments: "This is a test. This is another test" and "<b>This is a test. This is another test.</b>". The second segment is a bolded version of the first one. WorldServer correctly detects this repetition while TRADOS does not.

WorldServer also tolerates whitespace differences. Whitespace is ignored during 100% match generation and segments that differ only by whitespace are considered to be repetitions. WorldServer uses Java whitespace definition. See `Character.isWhitespace()` for more information.

Repetition calculations are greatly affected by segmentation rules. Smaller generated segments result in a higher chance of repeated fragments. TRADOS by default breaks sentences on the colon (`:`) character. This configuration allows for smaller segments and higher repetition counts, but might produce linguistically incorrect segmentation. WorldServer does not break English sentences on the colon character by default but can be configured to do so if required by a customer.

Scoping Test Results

SDL performed a test to compare word counts and repetition generated by TRADOS and WorldServer. A set of HTML files was scoped by the two products. A total of 11,000 files containing 60MB of HTML data were processed.

The results of the word counting test (total number of words counted) are shown in the following table:

TRADOS	**WorldServer Default**	**WorldServer Default + Number Counting**
4,436,009	4,395,495 (-0.91%)	4,445,387 (0.21%)

Using the TRADOS number-counting scheme, the WorldServer total number becomes 0.21% higher. This difference is attributed to the differences in word breaking described in this section.

The results of the repetition counting test (repeated words count percentage from total number of words) are shown in the following table:

TRADOS	**WorldServer**	**WorldServer + sentence breaking on colon**
13.5%	11.9%	13.9%

Matching WorldServer's sentence breaking rule with TRADOS' results in a slightly better percentage of detected repetitions. For information on WorldServer sentence breaking, see *Sentence Breaking* on page 166.

Chapter 19

Word Stemming

Topics:

The translation memory (TM) and terminology database (TD) features are key technologies that companies use to reduce their translation costs. As the amount of content increases within both of these repositories, the savings in translation costs also increases. The extent to which the content can be leveraged directly affects the customer's return on investment (ROI). In an ongoing effort to increase the value of WorldServer to its customers, SDL continuously seeks to enhance both its TM and TD technologies. One enhancement is the use of stemming technologies to improve the fuzzy matching and fuzzy scoring processes.

What Is Stemming?

Stemming is the process of identifying or creating canonical forms (or roots) for words with the purpose of normalizing out certain differences. These canonical forms are referred to as stems. The primary goal of stemming is to identify words that are related. For instance, the word `cat` and `cats` are related, but one is the singular form of the word and the other is the plural form. While the words may translate differently, you would expect the translations to be very close. When you compare these two words to each other, you would not expect a full penalty to be assessed, such as you would if you were comparing two words such as `cat` and `dog`.

Stems allow hits to be found for different forms of the sought word. For instance, if you are searching for `cats`, then you might find all of the entries containing `cats`, but you would not naturally find the entries containing the singular form `cat`. However, if the search process operated on the stems of words, versus the true form, then you would get the additional hits. For example, the stem for both `cats` and `cat` would most likely be `cat` (depending on the implementation). Therefore, searches for either `cat` or `cats` would yield the same results.

The translation memory segment leverage search implementations uses a shingle-based[15] algorithm, where word runs are created, stored and used for match lookup. Consider the following illustration of translation memory shingles created for the following two sample sentence fragments:

```
The cat in the hat
```

Generated shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

```
The cats in the hats
```

Generated shingles: `the|cats|in|`, `cats|in|the|`, `in|the|hats|`, `the|hats|the`, and `hats|the|cats|`

These shingles are used for finding match candidates based on the number of shared shingles. For the TM, this is used in the fuzzy matching process, where the candidates with the most shingles in common to the search string are returned. If we consider the two sentences above, and the shingles that would currently result, we notice that even though the sentences are close to each other in relation to the words being used, these two sentences do not share any common shingles. As a result, one would not be returned as a candidate for a possible fuzzy match for the other. If the sentence `The cat in the hat.` is stored in the TM, and `The cats in the hats.` is used for a new search, the search would result in no match candidates.

Stemming creates stems of each word, thus allowing certain variations in the words to be ignored. (that is, `cat` and `cats` would have the stem `cat`.) If the shingling process in WorldServer uses stemming, the above example would look more like this:

```
The cat in the hat
```

Generated shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

```
The cats in the hats
```

Generated shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

Notice that `cat` and `cats` would share the same stem, and `hat` and `hats` would share the same stem. As a result, all of the produced shingles for the two sentences would be the same. If we were to initiate a search using either of the sentences, the sentence would generate the same number of shingle hits to each of the strings. Both entries would be returned, and then scored. The resulting score would then reflect the actual differences between the two strings. If the sentence `The cat in the hat` is stored in the TM, and `The cats in the hats` is used for a new search, the search would result in a single match candidate. With stemming, the score would be a 60% match (reflecting the fact that 2 out of the 5 words in each sentence are different.)

[15] A shingle is a word run or a series of sequential words found in a reference string. In the TM, the shingles are generally 1–3 word runs, depending on the size of the TM entry string.

The results with stemming would produce more match candidates, thus potentially allowing greater leverage of the content stored in the repository. In this example, we were able to gain leverage of a 60% match, versus a 0% (or no match) without the use of stems. Arguably, the 60% score is too low in that it is based on treating `cat` and `cats` (and `hat` and `hats`) as completely different words. Perhaps instead of applying a full word penalty, a lower stemming penalty should be applied for different words that share a common stem. This lower penalty would reflect the fact that the words are considered to be related. Depending on the penalty value, the score of `The cat in the hat` when scored against a match of `The cats in the hats` could result in a score of 80%, 90%, 95% or whatever the user deems appropriate (provided that the penalty is configurable).

Between supporting stems and updating the scoring algorithm to treat commonly stemmed words differently, the ability to leverage the repositories can potentially be increased. The extent of the increase in leverage depends on the profile of the data. The more the content being translated contains slight variations in the word forms, the greater the increase in leverage will be from using stems and stem-based scoring.

Note: This potential increase does not come without a potential penalty. Stemming will effectively increase the number of match candidates having the same frequency of hits. Because the stems by their nature mask certain differences in the actual words used in the sentences, the differences cannot be identified until the scoring process. To ensure that the best candidates are being scored, more candidates need to be retrieved from the “hit frequency-based” candidate selection process. In short, this means more results may need to be retrieved and processed from the database. This may lead to more I/O and in-memory processing time spent for each segment that is leveraged. WorldServer allows this throughput to be configured, and so the customer can choose how to adjust this value. For some customers, increasing the throughput even when using stems may not significantly affect leverage levels.

Word Stemmer Support Framework

WorldServer allows you to create and configure stemmers to be used by WorldServer translation memory lookup operations. The following summarizes the word stemmer support:

- Supports an SDK Stemmer Component to allow you to develop and upload custom stemming implementation.
- Supports stemmer use in locating TM fuzzy matches.
- Provides the ability to associate stemmers with a specific language.
- Allows stemmers to be associated with specific TMs.
- Restricts a TM from being assigned multiple stemmers for the same language.
- Allows use of assigned stemmers to be enabled or disabled for specific TM.
- Generates standard shingles if stem use is disabled, or if no stemmer is assigned for the specific locale (whether source or target).
- Supports a configurable stemming penalty to be applied during the scoring process to denote word mismatches that have common stems. The assumption is that the penalty would not be as substantial as a completely different word.
- Supports an option that would allow both standard and stem-based shingles to be generated whenever a stemmer is provided. This would generate larger shingle tables, but would generate additional hits for tighter matches when stems are used.

The purpose of the stemmer is to generate the stem for a supplied word. The algorithm or heuristics to be employed should be language sensitive. A stemmer should be implemented for a specific language. To handle multiple languages, multiple stemmers must be provided.

Stemmers are usable during *ad hoc* search operations and internal search operations (such as the leverage process). When attempting to decide which stemmers to implement, it is useful to note that WorldServer leverage and scoping processes only perform searches from the source language. As a result, only the source languages require stemming implementations to affect scoping results. All additional language stemmers will be used only during the *ad hoc* search processes when the user does a target-based search. Therefore, creating stemmers for languages that do not represent source languages will add limited value to a WorldServer environment.

Using Word Stemmers with Translation Memory

Custom stemming implementations must be created and deployed before stemmer options are exposed in WorldServer. (Please refer to the WorldServer SDK documentation for details on creating and deploying custom stemmer implementations.) Once stemmer implementations have been deployed, you can configure your translation memories to use them.

The following section will describe the implications of using stemmers with translation memories. See the WorldServer *User Guide* for details on configuring options within the WorldServer UI.

Enabling/Disabling Stemming

The TM configuration screen provides the option to use stemming on a TM-by-TM basis. If the option is enabled, then subsequent storage and lookup processes will generate, store, and use stem-based shingles for the specified TM for the languages that have been configured to use stemmers. If the option is not selected, then standard shingles will be generated.

Assignment of Stemmers to TMs

To enable stemming, stemmers need to be associated with the TM. With appropriate permissions, customers can assign stemmers to, and remove stemmers from, any TM at any time. As new languages are supported, customers can update the TM configuration to use new stemmers.

When creating or configuring a TM, the user can choose from a list of available stemmers. The user can choose a particular stemmer and language combination. Even if a stemmer can support multiple languages, the user can select for which languages to use a particular stemmer. The list of options is generated by adding a selection option for each stemmer/language pair based on the languages that the stemmer reports being able to support. (Note that this can be done regardless of whether the stemmer reports locale objects or the actual language values.)

The system prevents the user from adding conflicting entries. Only one stemmer can be associated with the TM for a given language. To change the stemmer for a given language, the current stemmer entry must be removed.

The material in this section also applies to terminology databases.

Generating Standard and Stem Shingles

Searches based on standard shingles are sensitive to the slightest change in a word. They are optimal for finding exact matches and fuzzy matches with identical word runs. The higher the number of hits generated, the more likely the match is the one for which you are looking. Searches based on stemmed shingles are less sensitive to changes in words. Matches generating the same number of hits are less likely to be equally appropriate because certain (pre-scoring) differences are ignored[16]. Stem-based matching is geared more toward finding pure fuzzy matches, as opposed to finding exact matches.

Consider the examples discussed in the overview:

```
The cat in the hat
```

Standard shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

Stemmed shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

```
The cats in the hats
```

Standard shingles: `the|cats|in|`, `cats|in|the|`, `in|the|hats|`, `the|hats|the`, and `hats|the|cats|`

[16] The TM fuzzy lookup process has two phases. The first phase is to find match candidates based on hit frequency. The second phase is to score and rank, or in the case of the TD, validate the candidates.

Stemmed shingles: `the|cat|in|`, `cat|in|the|`, `in|the|hat|`, `the|hat|the`, and `hat|the|cat|`

The candidate selection process for the TM fuzzy lookup process returns the best candidates based on shingle hit frequency. Note the following cases:

1. If we are using standard shingles, we note that `The cat in the hat` and `The cats in the hats` produce a completely different set of shingles. As a result, a search for one of the sentences would never return the other sentence as a match candidate. This represents a potential loss in a high fuzzy match opportunity.
2. If we are using stemmed shingles, both sentences generate the exact same shingles. As a result, a search for one of the sentences returns the other sentence as a match candidate. Also, because the shingles are identical, the non-exact matching sentence results in the same number of hits as the exact match shingle. If the number of candidates returned from the database is restricted (as they are), it is possible that the best match would not be returned depending on how many equal hit candidates exist in the database and how many are pulled back for processing. (This is already possible, and stemming increases its likeliness.) To compensate, the allowed number of candidates during the candidate selection process may need to be increased.

   Only a subset of 100% matches rely on the fuzzy lookup process to be found. The leverage process searches for ICE and exact matches before even engaging the fuzzy lookup process. The exact match lookup requires everything to be exactly the same. It is possible to have 100% matches that are missed by the exact match lookup process. These typically include those entries that have certain textual differences that are not penalized (such as extra whitespace). As a result, increasing the allowed number candidate selections might not make an appreciable difference to the leverage results. (This is a good thing.)
3. If we are using both standard and stemmed shingles, then we will get hits against both sentences regardless of the sentence we use for our search. However, the more appropriate match will have more hits.

   In the above example, each sentence will generate five additional hits for the exact match versus the fuzzy match. This gives you the best of both options, but it comes with a price. This option requires the generation of two sets of shingles for every stored translation. That means that the shingle tables for the TM grows twice as fast, including the size of the table indexes! The additional space consumption coupled with the decrease in storage performance may outweigh the leverage benefits.

Having the `Generate both standard and stemmed shingle` option provides the customer with all of the above possibilities. Customers need to assess which configuration is optimal for their data profile.

This section mostly applies to terminology databases, though the TD does not have the same table size considerations described in Point 3. The TD would not double or even grow in the number of entries. Instead, it would need to support an additional search key field to allow for the storage of both standard and stem-based search keys.

Regenerate Shingles Based on Current TM Settings

The shingle generation process driven from the lookup process needs to be synchronized with the shingles that were generated during the TM entry storage process. Changes to the stemming configuration options can have the result that the saved shingles no longer match the shingles that will be generated for the search text (and for new TM entries added to the translation memory). In essence, this can happen when the TM is not empty and one of the following is true:

- the enabling or disabling of stemming has been toggled
- a stemmer has been assigned to a language that was not formally assigned a stemmer (and that language already contains entries in the TM)
- a stemmer has been changed or removed for a language already containing TM entries
- the stemming implementation has been updated

When the stored shingles are not synchronized with the current stemming configuration for the TM, leverage will most likely be negatively affected. (Note that this only applies to fuzzy matches within the TM, but to all standard matches within the TD.) As a result, the TM (and TD) configuration page presents the option to regenerate the shingles for the TM for the entries impacted. All processes accessing the TM or TD will be either blocked (by database locking) until the update is completed, or subject to leverage impact until the process is completed.

Common Stem–Word Difference Penalty

The use of stemming allows for the fuzzy process to identify some match candidates that it was previously unable to find. Words that share a common stem arguably should be scored in a manner that reflects their commonality. The current scoring algorithm does not take into account stem commonality between words being compared. Any change in the word results in a full word penalty being applied. For instance (as noted earlier), if the sentence `The cat in the hat` were compared to the sentence fragment `The cats in the hats`. the current scoring process would report a score of 60% because three of the five words are the same. `cat` and `cats` are treated as two completely different words.

The stemming penalty suggests that instead of treating words with a common stem as completely different, they should be treated as being related. Instead of applying a full penalty, only apply a partial penalty that reflects the perceived effort for translating from one form of a word to another of its form.

The TM property, `tm_score_same_stem_penalty`, allows you to configure this value. The value should be between `0` and `1`. A value of `0` means that no penalty is exacted on the match if the stem is the same, but the word is not exactly the same. A value of `1` means that the full penalty is taken, and different words with the same stems are treated as completely different words. A meaningful value is somewhere between these two extremes, and the customer should consider carefully what makes sense for them.

This penalty is applied only when stemmers are used. The default value for this penalty is `.05` (5% weighted penalty). Alternatively, it can default to `1`, which will force the customer to assign a meaningful value that may more appropriately represent the effort of translating from one form of a word to another.

The `tm.properties` file entry and default value is:

```
tm_score_same_stem_penalty=.05
```

Chapter 20

Translation Entry Locking and Auto Segment Locking

Topics:

WorldServer allows you to create projects that restrict updates to certain segments. The restriction is accomplished by using the segment locking feature within workflow steps where segment locking is enabled. The objective is to safeguard certain changes to translations made by language specialists (LSs), when those translation need to be protected. A typical process might look like the following:

1. A language specialist identifies a translation that needs to be corrected. To safeguard the change, the LS locks the segment. The LS must correct and lock each occurrence of the segment across all assets that contain it.
2. When the updated translation is saved to the TM, this lock status must be transferred to the TM.
3. When an asset is leveraged, the segment needs to be locked automatically if the match is an ICE or SPICE match and the TM entry is locked.
4. If the segment is unlocked explicitly (or programmatically), then saving it to the TM should remove the lock from the associated TM entry.

Locking the segment in the actual asset prevents the segment from being updated while the lock status is maintained. However, all lock statuses are lost if the asset is updated or if the cache is cleared for the asset. WorldServer now allows for translation entries to be locked explicitly and automatically. Similarly, asset segments can be locked automatically during the leverage process, thus potentially restoring user-defined locks even when the asset has been modified or the segment cache has been cleared.

Locking Translation Entries

Translation entries contain a lock status. You can set the status explicitly by using the TM entry edit tool and selecting or clearing the lock translation option. The TM entry can be locked automatically by locking an updated segment and saving the translation to the translation memory. When a new TM entry is created , it inherits the lock status from the segment that was used to create the TM entry. While you can lock at TM entry by locking a segment, you cannot unlock a TM entry by unlocking the segment. Additionally, the ability to lock a TM entry by locking a segment can be configured using the `enable_tm_entry_locking_for_locked_segments` TM property. This property is enabled by default. By setting it to false, segment locks will not be transferred to the TM entry.

Segment Auto-Locking

Asset segments can be locked directly. Locking or unlocking segments directly requires opening an asset in a WorldServer translation workbench via a human step that has step locking enabled. With the support of translation locking, asset segments can be locked automatically when ICE leveraged against locked TM entries. Users cannot modify the translation within the browser workbench if a segment is locked. Depending on the available options, the user might be able to unlock the segment. Only when the segment is unlocked can the translation be updated.

Segment auto-locking does not happen automatically. It requires the specific auto-segment locking condition to be enabled. The following configuration options are all set to `false` by default:

- If a SPICE match results and the TM entry is locked. Configure the TM property:

  ```
  enable_auto_segment_lock_on_locked_spice_match= false|true
  ```

- If an ICE match results and TM entry is locked. Configure the TM property:

  ```
  enable_auto_segment_lock_on_locked_ice_match= false|true
  ```

- ICE match with locked TM entry from same asset. Configure the TM property:

  ```
  enable_auto_segment_lock_on_locked_ice_asset_match= false|true
  ```

Note: Segments cannot be locked or unlocked explicitly in *ad hoc* mode. However, auto locking will take effect in *ad hoc* mode. Segments that are locked based on the auto locking mechanisms will appear to be locked in a WorldServer translation workbench, and cannot be edited.

Summary of Support

- Locking an updated segment and saving the translation to the TM results in the TM entry being locked.
- Unlocking the segment and updating the translation unlocks the TM entry when the translation is saved to the TM.
- Leveraging a locked TM entry results in the asset segment being locked when the match results in an SP/ICE match depending on the auto locking option enabled. In WorldServer 8.0, there are three new TM configuration options for controlling segment auto locking. They are all set to `false` by default.

Chapter 21

Translation Memory Best Practices

Topics:

Recent updates have been made to the translation memory architecture, geared at improving both performance and scalability. To benefit from these enhancements as well as general translation memory usage, the following best practices are encouraged.

Bilingual vs. Multilingual Translation Memories

WorldServer supports both bilingual and multilingual TMs. SDL recommends that, for performance reasons, each translation memory be bilingual—that it contain a single source and single target language. However, there management advantages in using multilingual TMs, in some cases.

Using separate TMs for each target language is already a common practice for many customers. Even though this is the recommended practice, WorldServer continues to support all previous multilingual configurations, and even the other configurations may experience performance enhancements for various operations when compared to environments earlier than WorldServer 8.0.

Some advantages of using bilingual TMs are as follows:

- Optimal performance of WorldServer translation memory architecture for critical translation memory operations. This configuration minimizes the performance impact that one translation memory has on another translation memory. The data is now separated at the database level for different user-defined translation memories.
- Optimal scalability of the WorldServer translation memory architecture. New translation memories can be added to the system without inheriting sizing limitations previously introduced by the presence of other translation memories.
- Elimination of most, if not all, currency issues associated with operations that require database level locks. Using bilingual translation memories will prevent all known potential concurrency risk points.

Some advantages of using multilingual TMs are as follows:

- You gain certain TM management advantageous because settings made when you create the TM apply to all language pairs. For example, if you set Allow Reverse Leverage, Attribute Masks, or ACLs on the TM, the setting applies to all language pairs, and can be adjusted for all pairs at once. Similarly, if you employ TM groups, you can set the TM penalty for each TM in the group, and can therefore control the penalty for all language pairs in the TM at the TM level. If each language pair had its own TM, you would have to create and maintain your settings in more TMs.
- You can search multiple languages without changing TMs. For example, if you had two related target languages, like Danish and Swedish, in the same TM, and you didn't find a match in one language, you might look in the other language to see if a close translation appears there for a hint.

  Note: TM groups accommodate a multilingual search, but you cannot perform deletes or search-and-replace operations when using a TM group.

In summary, bilingual TMs offer better performance, but multilingual TMs offer some management overhead benefits. If you need assistance deciding which strategy to use, please contact WorldServer Technical Support.

Purging Translation Memories versus Deleting Translation Memories

You should delete a translation memory only when you are no longer concerned about the data contained within it, and you no longer need to translation memory itself. If you intend to recreate the translation memory and restore its AIS associations, SDL recommends that you purge the translation memory instead. This will prevent you from having to create a new translation memory and prevents the need to restore AIS property associations.

Importing TMX Files as Background Jobs

Importing jobs in the foreground can lead to high resource contention and result in bringing the WorldServer environment down when too many import processes are running. SDL recommends that import jobs be run as background processes. This will allow them to be run in a prioritized queue that should be configured to prevent resource thrashing.

Back Up Translation Memories Regularly

Because accidents do happen as a part of normal translation processing, SDL recommends that you back up your translation memory data regularly and as often as can be supported. The recommended method is to back up the entire WorldServer database. If this is not feasible, you should at least back up regular exports of your translation data.

Repairing TM Entry Linkage after Moving Assets or Implementing Path Normalization

TM entries created within WorldServer are associated with the segment and asset that led to its creation. This linkage is important in establishing and maintaining additional content context. The context can be used to influence ICE and exact match ranking, to provide additional ICE context requirements, or to penalize matches applied to a different asset context. Depending on the use of this information, maintaining this information can be critical.

If translated content is moved, the TM entries that were previously created from it will be orphaned. Further translations will be added to the TM for the content instead of updating the existing TM entries, since the new location of the content will result in the assets being treated as new or different assets. As a result, the movement of content within the AIS system is discouraged, and should only be done if absolutely necessary.

The use of path normalization that is introduced at some point after the initial setup of WorldServer can result in a similar issue in that it affects the TM's perception of where the asset actually exists. Processing the same files (stored in the same AIS location) after path normalization has been implemented may lead to the TM to see the assets as being located somewhere new (as determined by the normalization implementation.) The end result is that the previous TM entries are now orphaned from the content they were originally associated.

While not ideal, one or both of the above scenarios may happen. The process below, while involved, provides a means for restoring the linkage of the assets to their orphaned TM entries. In short, the solution is to export the entries for each affected asset into a TMX file, update the paths globally in the TMX file, delete the original entries, and upload the modified TMX.

1. **Identify the impacted TM entries.** Using the TM search tools, you can search for the entries associated with a specific asset or a collection of assets sharing a common path by constructing an advance query against the “Entry of Origin” field.
2. **Export the selected TM entries.** The results of step #1 should be exported twice. One file will be keep for backup purposes, while the other will be updated.
3. **Update the paths in the TMX file.** The TMX file to be updated can then be edited using any text editor of choice. You may be able to easily use the editor’s search and replace tools to effective make the necessary file path changes. For instance, consider the following TM entry stored in a TM file:

   ```
   <tu srclang="en-US" creationid="Admin"
   creationdate="20070726T205127Z" changeid="Admin"
   changedate="20070726T205127Z" tuid="23-0">
   <prop type="x-idiom-source-ipath">/product1/release1/myFile.html</prop>
   <prop type="x-idiom-target-ipath">/product1/release1/myFile.html</prop>
   <tuv xml:lang="en-US">
   <seg>Segment Text</seg>
   </tuv>
   ...
   ```

   As a result of the move or path normalization process, the **/release1** portion of the path is no longer a part of the path. You may be able to search for all occurrences and replace it with “” (nothing).
4. **Import the updated TM entries into a test TM.** The process of importing the TM entries will associate the entries with the asset identified through the source and target paths stored in the TMX file entries.

5. **Verify that the linkages are in fact properly established.** To do this, take one of the assets that should be linked to the TM entries and associate it with the test TM. Record the count of TM entries imported into the TM after the import. Next perform a mock translation of the asset. Save and update the TM. Search the test TM for the entries associated with that asset and verify that the TM entries have been updated with the previous entries now designated as history records. The number TM entries in the TM should not have increased. Ideally, you would run all the assets through a mock translation process, and verify that the number of TM entries remains constant.
6. **Delete the original TM entries from the production TM.** Provided that the test in step #5 passes, delete the set of TM entries identified in step #1.
7. **Import the updated TM entries into the production TM.** Import the updated TMX file from step #3.

Appendix A

TM Properties

Topics:

This appendix gives details about the TM properties.

Configuring TM Properties

The WorldServer translation memory (TM) contains dozens of options that allow configuration for customer needs. The options are currently not exposed in the WorldServer UI. Instead, they are configured in a special TM properties file named `tm.properties`. WorldServer provides default values for all properties. These are hard-coded internally.

The `tm.properties` file is not required, unless you want to override any of the default values. To override default values, follow these steps:

1. Locate the file `tm.properties`.

   The file resides in the same directory as other WorldServer properties files:

   ```
   <WorldServer Installation Directory>\web-inf\classes\config
   ```

2. Open `tm.properties` for editing.
3. Find the property name-value pairs to override default values. The basic pattern is:

   ```
   [Property Name]=[Desired Value].
   ```

4. Save changes and start or restart WorldServer. The values from the `tm.properties` file are read during the WorldServer initialization process only. New values are not used until WorldServer is restarted.

Summary of Configurable TM Properties

The following tables summarize the TM properties that are supported by WorldServer.

AIS-TM Attribute Mapping

Property Name	**Description**	**Default Value**
`map_ais_property_to_same_named_tm_attribute`	Determines if the value for an AIS property is used to populate the value of a TM attribute with the same name. Effectively, AIS Properties are mapped to TM Attributes based on their names.	FALSE
`rank_based_on_ais_tm_attribute_match`	Determines if AIS-TM mapped attributes are used to rank match candidates. This includes both ICE and Exact match candidates. This requires that the map_ais_property_to_same_named_tm_attribute option be enabled.	FALSE

TM Entry and Segment Locking

Property Name	**Description**	**Default Value**
`enable_tm_entry_locking_for_locked_segments`	When set to TRUE, the TM entry is locked when you lock a segment.	TRUE
`enable_auto_segment_lock_on_locked_spice_match`	Determines if asset segments should be locked automatically when the TM	FALSE

Property Name	Description	Default Value
	entry of the match is locked and results in a SPICE match.	
`enable_auto_segment_lock_on_locked_ice_match`	Determines if asset segments should be locked automatically when the TM entry of the match is locked and results in a ICE match.	FALSE
`enable_auto_segment_lock_on_locked_ice_asset_match`	Determines if asset segments should be locked automatically when the TM entry of the match is locked, the TM entry is from the asset being leveraged, and results in an ICE match.	FALSE

Auto-Translation

Property Name	Description	Default Value
`enable_auto_translate`	Determines whether auto translation support is enabled. If set to true, the configured auto translation options are allowed. WorldServer provides a core implementation for auto-translation. Alternatively, customers can implement their own strategies via the SDK support of TM Services.	TRUE
`auto_translate_mid_variable_markers`	Set the characters that can be inside the variable segment . For example, ":-=" allows the following to be set as variables: SNN:123-35-6565 or ASD=123. Requires enable_auto_translate to be enabled in order to have any impact..	
`auto_translate_variables`	Determines whether to perform auto translation of number-based labels. For example, SN:234-35-6456 123ABCD AAA1232324. Requires enable_auto_translate to be enabled in order to have any impact.	TRUE
`min_letters_in_word` **Note:** The default changed in WorldServer 9.0 from `3` to `0` to accommodate Asian languages like Japanese and Chinese, which all have one-character words. For these languages, if this value were `3`, all words would be auto-translated.	Defines the minimum number of letters in a word for auto-translation purposes (does not affect scoping). Requires enable_auto_translate to be enabled in order to have any impact.	0

Controlling ICE Matches

Property Name	Description	Default Value
`enable_standard_ice_lookup`	Determines whether standard ICE matching will be supported. If you set this to FALSE, you effectively disable the standard ICE mechanism and operate solely on the SPICE matches. For instance, this might be desirable when processing Java property files. The option only affects the leverage process. ICE lookup methods in the SDK will still return standard ICE matches.	true
`require_asset_match_for_ice`	Determines whether the ICE condition will require that the TM entry match the TM AIS context of the asset being translated. If this option is enabled, then no segments for new assets will be ICE matched. Note that path normalization will affect how the TM identifies assets, and therefore can affect ICE results when this option is enabled. This option does not apply to SPICE matches. Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	false
`require_metadata_match_for_ice`	Determines whether the ICE condition will require the TM entry to have the same attribute values as the corresponding mapped AIS properties. If enabled, all mapped attributes much match their AIS property counterparts. This option does not affect SPICE matches. Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	false
`require_full_usage_context_for_ice`	Determines whether a full usage context is required for all ICE matches. For a full usage context, the TM match must have been translated between segments having the same content as those around the asset segment being translated. This option does not affect SPICE matches. Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	false
`require_reviewed_status_for_ice`	Determines if translation entries must have Reviewed translation status as a requirement for ICE matches. The default behavior is that unreviewed translation memory entries do not satisfy the ICE criteria. In non-live mode, all translation memory entries are reviewed, so ICE matches are unaffected by this configuration. This option should affect	true

Property Name	Description	Default Value
	both SPICE and ICE matches. Requires the enable_standard_ice_lookup or enable_sid_lookup option to be enabled in order to have an impact.	
`require_non_null_context_for_partial_ice`	Determines whether a null context should be enough to register a partial ICE match. This does not affect full context defined by a previous and next segment being null, as is the case for single segment documents. Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	true
`require_asset_match_for_single_segment_asset_ice`	Determines whether the ICE condition will require that the TM entry match the TM AIS context of the asset being translated when the asset only contains a single segment. If this option is enabled, then a single segment asset can only be ICEd if it has been previously translated. Note that path normalization will impact how the TM identifies assets, and therefore can impact ICE results when this option is enabled. This option does not apply to SPICE matches. Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	true

Leverage

Property Name	Description	Default Value
`enable_sid_based_alignment`	Determines whether SIDs are used during the alignment process if they are available. SIDs must match in order for the segments to be aligned if both segments contain SID values.	TRUE
`enable_sid_lookup`	Determines whether to leverage available SIDs for SPICE matches.	TRUE
`prefer_match_from_previously_matched_asset`	Determines whether the matching algorithm should prefer matches from an asset that provided a previous match, provided that the position of the segment in both assets matches. This option seeks to prefer pulling matches from a common source. In most cases, this option operates with minimal confusion. However, it can cause strange results when used in	TRUE

Property Name	Description	Default Value
	sub-optimal TM group configurations.	
`prefer_sid_over_ice_matches`	Determines whether SID matches should be a higher priority than ICE. If set to true, a SID Preferred ICE (SPICE) match is sought before looking up standard ICE matches. Otherwise, the ICE match is sought before looking for a SPICE match. If all content contains a SID, the impact of this setting is minimal. Similarly, if no content contains SIDs, then this property is fairly useless as well. This option has an impact for customer data that is a mix of SID and non-SID based content. Preferring SPICE matches over ICE matches suggests that global (SID) definitions should be preferred over the more localized context as defined for ICE matches. Setting the value to false suggests that SPICE matches should be used in the absence of a more localized context match. (ICE matching derives its context from the asset and surrounding segments, while SPICE matching derives its context solely from the segment identifier (SID).) This option is relevant only when both SPICE and ICE matches are being sought. Requires enable_standard_ice_lookup and enable_sid_lookup options both to be enabled in order for this option to have an impact.	TRUE

Performance

Property Name	Description	Default Value
`enable_iceplus_lookup`	No longer supported.	TRUE
`lookup_unlocked_ice_matches`	Determine whether unlocked ICE-match segments should be looked up again when a user selects re-apply TM or pretranslate. Generally, it is not worthwhile to set this option to true. It generally results in extraneous processing by the system only to yield the same results. This might be useful in cases involving an updated TM or changed TM. However, the same can be forced by simply	FALSE

Property Name	**Description**	**Default Value**
	choosing to have the asset re-segmented (though this will take more time). Requires enable_standard_ice_lookup option to be enabled in order to have any impact.	
`max_bulk_remove_per_query`	Defines bulk remove batch size.	500
`max_bwb_results`	The maximum number of fuzzy results requested by the Browser Workbench and export code. This number does not guarantee that this number of results will be generated. It controls the number of candidates that are retrieved and processed from the database. The candidate selection is based solely on the words in the segments. As a result, the fuzzy candidates retrieved may not be the best depending on potential differences associated with punctuation, numbers and placeholders. Increasing this value increases the likeliness of retrieving optimal fuzzy matches. However, increasing this value also affects performance of the associated process. This value affects fuzzy match lookup, and does not affect ICE or exact match lookup.	20
`max_fuzzy_results_in_asset_lookup`	The maximum number of fuzzy match candidates that are returned from the database. A SQL query does some coarse filtering to pull a manageable number of entries from the database, and then those candidates are further filtered and scored. This constant controls the size of the candidate list that is returned from the database when doing a full document lookup. Increasing the value increases the likeliness of retrieving or scoring the best fuzzy matches. However, it will also increase the amount of data that needs to be retrieved from the database and processed by WorldServer. (This property is similar to max_bwb_results, but it is applied to the leverage process.) This value potentially affects the fuzzy scoping results.	5
`max_search_results`	Determines the default value for the TM search results, and for a number of other related processes (including search and replace). For the TM search, it has been exposed to allow the user to override this value. For the other processes, like search and replace, this value represents the maximum number of search results.	5000
`shingle_excluded_characters`	Determines which characters to exclude from shingle tables. Currently excludes excessively common characters such as comma, dot, colon, and latin comma, latin dot, latin colon. It enables the customer to exclude optimized TM search and shingling support for common non-alpha characters for performance reasons; users can configure which characters fall into this category.	`,.:/_-`

Property Name	Description	Default Value
`shingle_index_tablespace`	Controls index table space for shingle tables. WorldServer dynamically creates what it refers to as shingle tables for each new TM database configured in WorldServer. These tables are required for the fuzzy matching lookup processes. These tables can grow significantly as the TM grows, and it might be useful to require that they be created in their own database tablespace. The specified tablespace must exist. WorldServer does not create the specified tablespace. Once supplied, WorldServer creates all new shingle tables in this tablespace. Do not change this optionafter system setup because it will not be able to find the previously created shingles tables. While this should not create an error condition for WorldServer, it prevents WorldServer from gaining access to and using the old shingle tables, which would result in a loss of all fuzzy matches until the entries are regenerated.	

Repair

Property Name	Description	Attributes	Default Value
`prevent_all_segment_repairs`	Determines whether all repairs are disabled. Setting this value to true disables all TM repairs regardless of their individual settings. Setting it to false disables the option, thus allowing repairs to executed based on their default or configured settings.	N/A	FALSE
`capitalization`	This repair attempts to ensure capitalization consistency of replaced text to that of the original text. This works in conjunction with the word transform repair. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true`	all and true
`min_unrepaired_score_for_upgrade`	The minimum unrepaired score for a match that can be upgraded to a 100% score. The upgrade occurs only when the repaired score is 100%. If the unrepaired score is less than this value, then it will never be upgraded to a 100% match.	N/A	0.8

Property Name	**Description**	**Attributes**	**Default Value**
	This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.		
`number_deletion`	This repair attempts to delete extra numbers from the TM match text. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true`	all and true
`number_transform`	This repair attempts to handle number differences between the segment text and the TM match text. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true`	all and true
`placeholder`	This repair attempts to address formatting differences between the segment text and the corresponding TM match. It allows a penalty to be assessed against placeholder repairs that WorldServer does not consider safe. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true` `unsafe_repair_penalty=0.0`	0.0
`punctuation`	This repair attempts to align the outer punctuation of the TM match to that of the source segment. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true`	all and true
`word_deletion`	This repair attempts to delete extra words from the TM match text. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=true`	all and true

Property Name	**Description**	**Attributes**	**Default Value**
`word_transform`	The current implementation seeks to transform untranslated text in the source and target text of the match to match that of the source segment. Do not use this repair with languages that share common words. This option requires that the prevent_all_segment_repairs be disabled in order to have any impact.	`target_languages=all` `enabled=false`	false

Scoping

Property Name	**Description**	**Default Value**
`min_scoping_score`	Defines the minimal score to be registered for scoping purposes, which happens in conjunction with the segmentation and leverage processes. If fuzzy matches are sought, the score of the best match is recorded as long as it exceeds this value. Anything less than this value will not be recorded. If no matches exceed this value, than the segment is scoped as a 0% match. This value also defines the default minimal score for TM matches returned by the SDK TM match lookup process.	0.5

Match Scoring

Property Name	**Description**	**Category**	**Default Value**
`tm_score_capitalization_penalty`	Defines the penalty to assess for each capitalization difference. This value does not necessarily represent a final score penalty because it is weighted by the number of occurrences.	Content Level	0.01
`tm_score_extra_placeholder_penalty`	This value determines whether placeholders should be considered in the scoring process. It does not actually represent the applied penalty. There are two other properties that are reserved for this purpose. If the value is 0, then placeholders are omitted from the scoring process. Setting the value to a number greater than 0 results in the placeholders being used to affect the scoring. Setting this property to 0 is the same as setting both the tm_score_inner_placeholder_weight and	Content Level	1

Property Name	Description	Category	Default Value
	tm_score_outer_placeholder_weight values to 0.		
`tm_score_inner_placeholder_weight`	Defines the inner placeholder tag weight (word weight is 1.0). Allows the effective weight of embedded placeholders to be set. The effort in correcting placeholder placement may not be the same as correcting a word. This property allows the weight to be set to establish a relative weight to a word element.	Element weight	0.25
`tm_score_number_difference_penalty`	Defines the number difference penalty. This value does not necessarily represent a final score penalty as it is weighted by the number of occurrences.	Content Level	0.2
`tm_score_number_weight`	Defines the number weight (word weight is 1.0). Allows the effective weight of numbers to be set. The effort in correcting numbers may not be the same as correcting a word. This property allows the weight be to set to establish a relative weight to a word element.	Element weight	0.2
`tm_score_outer_placeholder_weight`	Defines the outer placeholder weight (word weight is 1.0). Allows the effective weight of outer placeholders to be set. The effort in correcting placeholder placement may not be the same as correcting a word. This property allows the weight be to set to establish a relative weight to a word element.	Element weight	0.1
`tm_score_placeholder_sequence_penalty`	Defines the penalty for non-matching sequence numbers. This value does not necessarily represent a final score penalty as it is weighted by the number of occurrences.	N/A	0.01
`tm_score_punctuation_penalty`	Defines the penalty for punctuation/non-alphanumeric character differences. This value does not necessarily represent a final score penalty as it is weighted by the number of occurrences.	Content Level	0.005
`tm_score_same_stem_penalty`	Defines the penalty to be applied when the words being compared are different, but have the same stem. This property is used only when word stemmers are supported. Words with	Content Level	.2

Property Name	**Description**	**Category**	**Default Value**
	the same stem are generally related, and should be treated as related.		
`tm_score_asset_mismatch_penalty`	This penalty allows you to penalize a TM match if it is not associated with the asset that is being translated. When assets are translated, matches are created that associated with that asset. As a result, the next time the asset is translated, WorldServer is able to bias the selection of matches based on whether they are associated with the newer version of the asset. If this penalty is set to a value greater than zero, then the leverage process will also penalize matches that are not associated with the asset being translated. Take care when using this penalty, as it affects all matches except SPICE matches. Nonetheless, this option may be useful to you if your quality requirements must take the source asset into consideration when evaluating ICE and exact matches. If your intentions are to prevent ICE matches against matches from a different asset, then you should use the ICE restricting option instead.	Leverage Level	0
`tm_score_metadata_mismatch_penalty`	This penalty allows you to penalize matches that do not have a complete set of matching values with the asset for all AIS-TM mapped attributes. Instead of using physically different TMs to partition translation data, metadata can be used. For instance, you can use a mapped attribute to track the product with which the TM entry is associated. If the products are significantly different in the terminology used or in the language, you might want to further restrict the extent to which translations are cross leveraged. The use of this penalty allows you to do that.	Leverage Level	0
`tm_score_multiple_exact_match_penalty`	This penalty allows you to penalize exact matches that occur when there are other exact matches for the segment with differing translations. Exact matches are ranked using the	Leverage Level	0

Property Name	Description	Category	Default Value
	same rules for ICE matches, and generally will provide the best available match. Nonetheless, perhaps the existence of multiple translations for a segment implies the introduction of a quality issue. There is already the option to detect multiple exact match segments; however, the target segment will still be populated with the system's best guess for the exact match. If you are not comfortable with WorldServer choosing the best exact match for you, you could choose to penalize such matches so that they would be available to the translator as high fuzzy matches (or whatever leverage level as dictated by the penalty) instead of them being positioned solely for review as an exact match. The use of this option requires that the view multiple exact matches option is enabled. This penalty does not affect ICE matches.		
`tm_score_whitespace_difference_penalty`	Defines the penalty to be applied when there are whitespace differences between the source text of the segment and the source text of the TM match. This is a content based penalty, not a leverage level penalty. Value range should be between 0 and 1.	Content Penalty	0
`tm_score_unreviewed_match_penalty`	This penalty is applied to a match that is being leveraged that does not have the “Reviewed” status. Depending on whether your WorldServer environment is using Live TM mode or not, you may have TM entries stored in your TM that do not have a reviewed status. Working in the non-Live TM mode, all matches stored to the TM are viewed. However, in Live TM mode, the stored TM entries may have various statuses, such as “Reviewed”, “Unreviewed” or “Rejected” depending on what is happening within the project. (Refer to the documentation on Live Translation Memory.)	Leverage Penalty	0

Property Name	**Description**	**Category**	**Default Value**
	This penalty is a good way to control how the leverage treats matches that do not have the “Reviewed” status. By default, non-reviewed matches will not result in ICE matches, but they may lead to 100% matches. This is generally all right, since 100% matches typically must be reviewed anyway. During the leverage process, matches resulting from these TM entries will lead to the target segment being auto populated because it is a 100% match. Perhaps your treatment of 100% matches is different and your goal does not require 100% matches to be reviewed. In order to do this, you need to ensure that the 100% matches that get registered meet some baseline quality guidelines. In this case, you might want to minimally require that only 100% matches that have been reviewed. This minimal goal can be accomplished by setting the Unreviewed penalty to a value greater than `0`. **Note:** As you review the other leverage level penalties, there may be other opportunities for you to eliminate lesser quality 100% matches so that you end up with 100% matches that may not require further review, depending on your quality requirements. This penalty affects all matches, including SPICE and ICE. If your intentions are to prevent ICE matches against unreviewed matches, then you should use the ICE restricting option instead.		
`tm_score_reverse_leverage_penalty`	This penalty allows you to penalize all reverse leverage matches. In WorldServer, TM content can be leveraged in both directions—source to target, and target to source. This is referred to as bi-directional leverage support. The maximum leverage level supported for reversed leverage	Leverage Level	0

Property Name	**Description**	**Category**	**Default Value**
	matches is 100%. If a reverse leverage match results in a 100% match, it is treated the same way a normal, forward leveraged match—it will be populated into the target segment. Depending on your quality metrics, this may or may not be acceptable. You may want them to only be treated as high fuzzy matches. If this is the case, then this penalty can be used to prevent reverse leverage matches from generating 100% matches, and thus, prevent the target segment from being auto populated with them during the leverage process.		
`maximum_target_length_penalty`	Defines the penalty to be applied to a hit with a translation that is too long	Leverage Level	.01
`sid_mismatch_penalty`	Determines penalty for SID mismatch condition. (`0.0-1.0`). Rules: • If both source and TM match have SID values, if different, then apply penalty • If neither have SID values, no penalty • If source has SID, and TM match does not, no penalty • If source has no SID, and target does, apply penalty	Leverage Level	0

Split/Merge

Property Name	**Description**	**Default Value**
`do_fuzzy_automatic_merge`	Determines whether automatic merging is attempted for fuzzy matches. For fuzzy matches, the objective is to combine asset segments so that they collectively yield a higher scoring match in the TM. Merging is only attempted when a suitable match has not been found for the current segment being processed. The do_100%_automatic_merge and do_ice_automatic_merge options must be enabled.	TRUE
`do_fuzzy_automatic_split`	Determines whether automatic splitting is attempted for fuzzy matches. The do_100%_automatic_split and do_ice_automatic_split options must be enabled.	FALSE
`do_hyper_merge`	Determines whether hyper (formatting insensitive) merging is attempted. This is similar to the auto merge options. However, this one can only be used in the fuzzy matching process. The	TRUE

Property Name	**Description**	**Default Value**
	standard merge is sensitive to placeholders (formatting). This merge allows for segments to be merge based on content alone. This merge will most likely not result in 100% matches, but it will result in higher fuzzy leveraging. It is designed to compensate for changes in segmentation rules (whether manual, migrated from other TMs or the result of filter configuration changes.) It also helps increase leverage across different document types where the number or types of formatting encodings may not be consistent. Merging is only attempted when a suitable match has not been found for the current segment being processed. The do_fuzzy_automatic_merge must be effectively enabled in order for this option to have any impact.	
`do_ice_automatic_merge`	Determines whether automatic merging is attempted for ICE matches. For the ICE option, merging attempts to restore ICE matches that have resulted from segments that have been manually merged and translated by a translator. Without this capability, WorldServer cannot restore the manually merged segment. Note that manually merging segments goes against the segmentation rules applied by the asset filters. Merging is only attempted when a suitable match has not been found for the current segment being processed. Requires either enable_standard_ice_lookup or enable_sid_look option to be enabled in order to have any impact.	TRUE
`do_ice_automatic_split`	Determines whether automatic splitting is attempted during ICE matching processes. If this is disabled, then all splits are disabled. If the split does not result in an ICE match, then it should be undone unless 100% splits option is enabled. Note that this option does not require that both parts of the split result in ICE matches. It is geared toward finding an ICE match for the first of the two new segments. Requires either enable_standard_ice_lookup or enable_sid_look option to be enabled in order to have any impact.	TRUE
`do_100%_automatic_split`	Enable splitting during 100% lookup. If this is disabled, then fuzzy splits are also disabled. If the split does not result in a 100% match, then it should be undone unless the fuzzy split option is enabled. Note that this option does not require that both parts of the split result in 100% matches. It is geared toward finding a 100% match for the first of the two new segments.	FALSE
`do_100%_automatic_merge`	Enable merges to produce 100% matches. If this is disabled, then all fuzzy merge options are also disabled. If a 100% or better match does not result, then all segments involved in the merge should be restored, unless the fuzzy merge option is enabled	TRUE

Storage

Property Name	**Description**	**Default Value**
`import_empty_translated_segments`	Determines whether translated segments containing only whitespace or placeholders can be imported into the TM from a TMX file. This system-wide option applies to new TM entries, and does not affect entries that have already been saved in the TM.	FALSE
`save_empty_translated_segments`	Determines whether translated segments containing only whitespace or placeholders can be stored in the TM when an asset is saved. This option applies system-wide to determine whether WorldServer allows TM entries to be stored that do not contain a translation. This option applies to new TM entries, and does not affect entries that have already been saved in the TM. Note that the `save_untranslated_segments` property applies only to Browser Workbench behavior. This option supersedes the Browser Workbench-specific option, and also applies to custom coded processes.	TRUE
`save_ice_match_segments`	Determines whether ICE matched segments are saved during the update TM process. If set to FALSE, then ICE matches are not saved. If set to FALSE, even if the ICE matches are associated with the current asset already, they still are not saved since they are already in the TM. Similarly, if set to TRUE, and the ICE matches are already associated with the current asset, they still are not saved since they are already in the TM. If set to TRUE, and the ICE matches were derived from a different asset, then a copy of the TM entry will be created and saved in association to the current asset segment. If TM groups are used, you may want to set this value to true to ensure that the entry is saved in the associated write TM. If the write TM already contains that (identical) entry, then it will not resave the entry. This value is FALSE by default.	FALSE
`save_untranslated_segments`	Determines whether untranslated segments in the Browser Workbench are saved in the TM as translations. Enable this property when you want WorldServer to save empty targets for segments that are intentionally not translated. Not saving to the TM allows the target to be generated without translating the given segment. However, if the asset is updated later, new TM entries that match the untranslated segment may be used. This option allows an empty translation to be stored and associated with this asset. Note that if an empty translation is stored, it becomes a possible candidate for future documents as well, which might or might not be desirable.	FALSE

User Interface Properties

Property Name	Description	Default Value
`enable_bwb_multiple_exact_matches`	In Browser Workbench, segments with multiple 100% candidates are now identified by a different border style, featuring double solid blue lines. By default, this feature is disabled because it affects performance during 100% matches.	FALSE
`bwb_display_updated_to_tm_as_ice`	Allow segments saved to the TM to show immediately as ICE matches in Browser Workbench. The solution involves supporting a configurable option since this change could negatively affect a customer's defined processes or experience.	FALSE
`tm_newline_separator_keyword`	Determines the keyword which is used for marking the new line separator. This is set to `[NL]` by default. This is the code to use in the Search and Replace tool to represent a new line character.	[NL]

Live TM Mode Properties

Property Name	Description	Default Value
`enable_live_translation_memory`	Determines if WorldServer is using Live Translation Memory. If the setting is off, then TM update behaves largely the same as in WorldServer 8.0. If the setting is on, then WorldServer TMs are updated whenever the segment cache is updated using translation statuses.	false
`set_translation_statuses`	Determines if segment translation statuses are set at leverage time based on statuses of TM matches. In live mode, this is always the behavior and there is no way to turn it off (this property setting is ignored). In non live mode, if this property is set to "true", then ICE matches lead to a segment status of Reviewed, and 100% matches lead to a segment status of Pending. In non-live mode, the default behavior is that segment translation status are not set based on TM matches, as in WorldServer 8.0.	false
`maximum_exact_translation_status`	The maximum segment translation status for 100% matches. By default, 100% matches lead to an asset segment translation status of "Pending Review".	Pending

Deprecated Properties

Property Name	**Description**	**Default Value**
`allow_punctuation_repairs`	Globally enables or disables the punctuation repair. This property was replaced with the locale-to-locale mapping implementation. (See the punctuation property above.)	TRUE
`allow_number_repairs`	(Never actually supported.)	N/A
`treat_whitespace_as_significant` **Replaced in WorldServer 9.0**	This property is no longer supported. You should remove it from your `tm.properties` file. (If it is included, you will be unable to start WorldServer.) The TRUE/FALSE `treat_whitespace_as_significant` property in `tm.properties` has been replaced by a new penalty property, `tm_score_whitespace_difference_penalty`. If you had set `treat_whitespace_as_significant` to `false`, you now want to set `tm_score_whitespace_difference_penalty` to 0. If you had set `treat_whitespace_as_significant` to `true`, you now want to set `tm_score_whitespace_difference_penalty` to the value to which you have set `tm_score_punctuation_penalty` .	N/A

Appendix B

Logging TM and Alignment Processes

Topics:

This appendix describes how you can use WoirldServer logging facilities to gather information that may help you diagnose TM and alignment issues.

Logging TM and Alignment Processes

If you are investigating TM or TM alignment-related issues, it is often useful to collect debugging information. To do this, enable the WorldServer debug logging for these processes.

TM Debug Logging

To enable general TM-related logging, add this entry to the `general.properties` file:

```
log4j.category.com.idiominc.ws.tm=debug
```

TMX Alignment and Asset Alignment Logging

To enable alignment-related logging, add this entry to the `general.properties` file:

```
log4j.category.com.idiominc.ws.autoalignment=debug
```

Configuring a Separate File for Collecting Debug Information

The TM- and alignment-related debug options result in a substantial amount of data being added to the WorldServer logs. Instead of writing the information to the standard WorldServer log file, consider configuring dedicated log files for each.

To do this, you can configure a new rolling file appender within the `general.properties` file, and using this appender to collect the debug data. This example shows how you would do this to capture alignment data in a separate file:

```
# enable WorldServer alignment debug messages
log4j.category.com.idiominc.ws.autoalignment=debug
# Create an Appender for logging alignment-engine debug messages
log4j.appender.alogfile.File=C:/logs/alignmentlog2.txt
log4j.appender.alogfile=org.apache.log4j.RollingFileAppender
log4j.appender.alogfile.MaxFileSize=100000KB
log4j.appender.alogfile.MaxBackupIndex=100
log4j.appender.alogfile.layout=org.apache.log4j.PatternLayout
log4j.appender.alogfile.layout.ConversionPattern=[%d]: %m%n
```

Appendix C

Sentence Breaking

Topics:

This appendix describes how the WorldServer segmentation process determines how to divide asset text into individual sentences.

Sentence Breaking

The WorldServer sentence breaker engine is rules based. It uses a set of regular expression based rules to decide where to break text into sentences, and rule exceptions to block the breaking rules in some cases. The engine uses the Perl regular expression classes from the Jakarta ORO package. For more information, consult the documentation for the `Perl5Util` class that is available at *http://jakarta.apache.org/oro/index.html*.

The rules are contained in a standard properties file named `SentenceBreaker.properties` located in the `WEB-INF/classes/config` directory inside the WAR file. WorldServer has separate instances of the engine for each language. The sentence breaker file contains base definitions for the sentence breaking rules. The language specific files can add new rules or override either parts of the rules or the complete rules set. For example, `SentenceBreaker_fr.properties` may contain additional rules and exceptions applicable only for French.

The rules are defined in the above mentioned property file. In addition to the rules themselves, the property file contains macros, which are recurrent pieces of regular expression definitions that can be used in rules.

Rule Definition

The basic structure of the property file is as follows:

1. A rules *list* is defined.
2. Each named rule in the list is defined.
3. A rule exception list and the exceptions themselves are specified.

The Rules List

A rules list is a comma separated list of rules names. The list of rules is specified by the property “rules,” followed by a comma separated list of rule names, each of which is a string. For example, here is a rules list containing the `break_on_dot` and `break_on_question_mark` rules:

```
rules=break_on_dot,break_on_question_mark
```

Adding the Named Rules

Each rule has four parts:

- A prefix pattern matching the text before a break
- A break pattern for the break itself
- A suffix pattern matching anything relevant after the break (in case it's necessary to include this in an exception)
- A list of named exceptions

The rule prefix pattern is defined by a property: `rule.prefix.<rule_name>`. The break pattern is defined by a property: `rule.break.<rule_name>`. The suffix pattern is defined by a property: `rule.suffix.<rule_name>`.

Here are examples of a prefix, break, and suffix patterns:

```
rule.prefix.break_on_dot =[^\\.]*   <- matches a span of non-dots
rule.break.break_on_dot =\\.[ ]*  <- matches the dot (.) and the remaining
whitespace
rule.suffix.break_on_dot =[^ ]*  <- matches any number of non-space characters
(next word)
```

Finally, a rule exception list and the exceptions themselves must be specified. Each exception is named. The list is defined as a comma separated list for `rule.exceptions.<rule_name>` property, for example:

```
rule.exceptions.break_on_dot=month_exception,title_exception
```

Each exception is defined by an exception source and a matching pattern. The exception source defines the text to be matched against, and the pattern is a regular expression to be used for the matching. If the match occurs, the breaking is disabled.

The source describes the string that is used to match the exception against. It is defined as a string almost certain to contain placeholders of the form `{prefix}`, `{break}`, and `{suffix}`. These placeholders will be replaced with the prefix, break, and suffix parts of the rule match as described above. Consider this example:

```
rule.exception.source.month_exception ={prefix}{break}
rule.exception.pattern.month_exception=(Jan|Feb|Mar|Apr|Jun|Jul|Aug|Sep|Sept|Oct|Nov|Dec)\\.
 *$
```

During execution, the string constructed from `rule.exception.source` is matched against `rules.exception.pattern`. If the match occurs, the breaking rule is suppressed.

Macros

A user can define a recurrent piece of a regular expression definition as a macro. The macro then can be used in other definitions throughout the file. The macro is defined as `macro.`*`<MACRO_NAME>`* and referred to as `:`*`<MACRO_NAME>`*`:`.

For example, the macros below define two characters for CJK punctuation, and then combine them into another macro:

```
macro.CJK_PERIOD=\u3002\ufe52\uff0e\uff61
macro.CJK_QUESTION_MARK=\u2049\ufe56\uff1f

macro.CJK_BREAK_CHARS=:CJK_PERIOD::CJK_QUESTION_MARK:
```

The new macro is used later on in a regular expression:

```
rule.prefix.cjk_break=[^:CJK_BREAK_CHARS:]*
```

Debugging

Exception evaluation can be made verbose by adding a debug flag in the form of the additional property: `rule.exception.debug.`*`<exception_name>`*`=true`. This flag causes the Sentence Breaker engine output exception evaluation detail like the following to the standard output:

```
++ exception name = month_exception ++
prefix = "He is smart"
break = ". "
suffix = "He"
matching string = "He is smart. " ({prefix}{break})
matching regexp = "(Jan|Feb|Mar|Apr|Jun|Jul|Aug|Sep|Sept|Oct|Nov|Dec)\. *$"
result          = false
++ exception name = month_exception ++
prefix = "He starts on Aug"
break = ". "
suffix = "25."
matching string = "He starts on Aug. " ({prefix}{break})
matching regexp = "(Jan|Feb|Mar|Apr|Jun|Jul|Aug|Sep|Sept|Oct|Nov|Dec)\. *$"
result          = true
++ exception name = month_exception ++
prefix = "25"
break = "."
suffix = ""
matching string = "25." ({prefix}{break})
matching regexp = "(Jan|Feb|Mar|Apr|Jun|Jul|Aug|Sep|Sept|Oct|Nov|Dec)\. *$"
result          = false
```

The output shows how the prefix, break and suffix components have been constructed, and how a matching string for the exception was generated. It also shows the regular expression being applied and the match result.

Index