How to prepare DITA content for localization

Contents

Chapter 1. Writing with localization in mind..... 1

Translation-friendly text..... 1

Non-translatable content..... 2

Glossaries..... 3

Localization-friendly images..... 5

Chapter 2. Preparing content in oXygen..... 6

Project structure..... 6

Preparing DITA code for localization..... 8

Text sorting and direction..... 9

Content reuse..... 10

Resolving conrefs and keyrefs..... 12

Chapter 3. Translating content in memoQ..... 15

Importing a DITA project..... 15

Translating repetitions..... 18

Chapter 4. Glossary..... 21

Chapter 1. Writing with localization in mind

Translation-friendly text

Learn a few tips to make your text easier to translate and less likely to cause translation errors.

Clarity

A text that is easy to understand is usually also easier to translate. To make your writing clearer, use these tips:

- Avoid overly long sentences and paragraphs.
- Repeat important words to avoid misunderstanding.

~~If the shaft of your wand breaks, you can get a new **one** online.~~

If the shaft of your wand breaks, you can get a new **wand** online.

- Don't describe more than one action in a sentence unless writing about strictly related actions.
- Use words such as "and," "then," "but," "a," "the," "this," and "that" to make your message clear.

Grammar

You can make your text easier to understand by using certain grammatical forms. To make your writing clearer, follow these rules:

- Express actions with verbs, not nouns.

~~Applying excessive force can lead to **rupture** of your wand.~~

If you apply excessive force, your wand can **break**.

- Use active voice.

- Split clusters of nouns into smaller logical units.

~~magic wand cleaning center~~

center for cleaning magic wands

Formatting

Search your text for unintended line breaks and punctuation marks (such as periods in place of commas). *CAT* tools may misinterpret such characters and split sentences into separate units. This can lead to translation errors and inconsistencies.

Avoid using page breaks and empty lines to set the layout of your document. The same content can have a different volume in different languages.

Important:
When localizing a DITA project, always send the translators your source files rather than the output (such as PDF or HTML files).

Related information

Localization-friendly images *(on page 5)*

Project structure *(on page 6)*

Non-translatable content

Learn about types of content that you may not want to translate.

Types of non-translatable content

While planning the localization of your project, it's important to identify fragments you don't want to translate. These can include the following:

- Proper names, such as brand names.
- Contact details (though you may need to translate some details, for example country names).
- UI text (if the UI has not been localized).
- Code blocks (but you may want to translate comments in the code).
- Legal text (you may need to handle it separately).

Tip:
You can use glossaries *(on page 3)* to let translators know which phrases they shouldn't translate.

UI text

If your project contains UI terms, it's important to know whether the UI has been translated, and into which languages.

If the UI hasn't been localized, you should let translators know how to handle UI text. For example, you may want them to follow each UI term with its translation in brackets.

Table 1. Example of handling UI text translation

English source	Polish translation
Click **New wand**.	Kliknij **New wand** (Nowa różdżka).

Related information

Preparing DITA code for localization *(on page 8)*

Localization-friendly images *(on page 5)*

Glossaries *(on page 3)*

Glossaries

Learn the benefits of using glossaries in your localization process.

Consistency

Glossaries are the easiest way to improve the consistency of translated text. Different translators (or even the same translator at different times) can translate the same term in various ways. Giving them a glossary of approved translations will make their translations more consistent.

Note:
Most *CAT* tools enable automatic terminology checks to make sure that a translator has used terms from the glossary. But remember that such checks are less reliable in languages where one word can have many grammatical forms.

Non-translatable text

Glossaries can also specify text that should not be translated, such as brand names.

Forbidden terms

If you don't want a given term to be translated in a particular way, a glossary is also a good place to specify that.

For example, the word "pacemaker" can be translated into Polish as "rozrusznik," but you may want translators to use the less colloquial "stymulator" instead.

Definitions and context

Glossaries can also provide a definition for each term, as well as context (such as subject domain or sample use).

Format

You can develop a glossary in a spreadsheet editor. Simply prepare a list of terms with their translations and optionally other details, such as definitions.

Once you finish, export your glossary as a *CSV* file. Translators can then import the file into their CAT tool.

Figure 1. A simple bilingual glossary developed in a spreadsheet editor

	A	B
1	**English**	**Polish**
2	curse	klątwa
3	magic	magia
4	magic wand	magiczna różdżka
5	spell	czar
6	spell book	księga czarów
7	wand	różdżka
8	wizard	czarodziej

Glossaries (called term bases) can albo be developed directly in a CAT tool. Most CAT tools have their own term base format, but you can use a universal format called TermBase eXchange (TBX) to move term bases between different tools.

Related information

Non-translatable content *(on page 2)*

Localization-friendly images

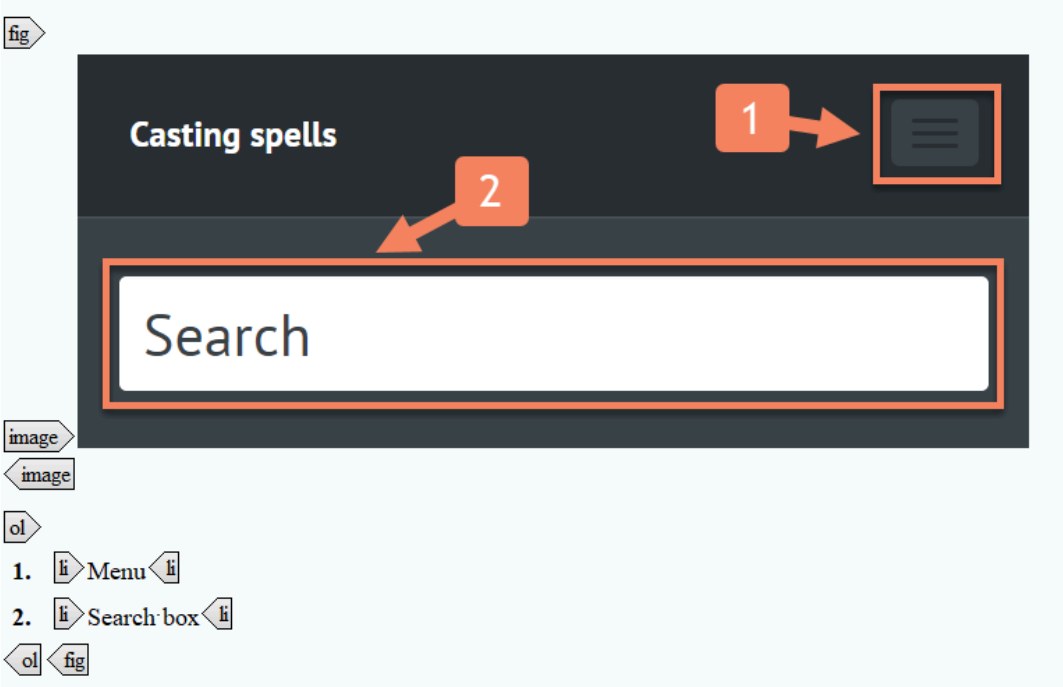
Learn a few tips to make your images easier to localize.

Image format

If your images contain text that you want to translate, avoid using bitmaps, such as JPG or PNG files. Use *SVG* files instead. Most *CAT* tools can translate text contained in SVG files.

Callouts

To simplify the localization process, avoid using descriptions within images. Instead, use numbered callouts and explain them under the image.



UI screens

If your documentation contains UI screens, it's important to consider whether the UI text *(on page 3)* has been localized. You may need to plan the additional step of capturing UI screens in each target language.

Related information

Translation-friendly text *(on page 1)*

Non-translatable content *(on page 2)*

Chapter 2. Preparing content in oXygen

Project structure

Learn how to structure your project to make localization easier.

Folder structure

A logical folder structure will make localization much easier. Don't keep all files in one folder. Instead, create separate folders for your topics, images, and other content.

> **Tip:**
> To avoid publishing problems, always store your DITA maps at the same level or above your topics and images.

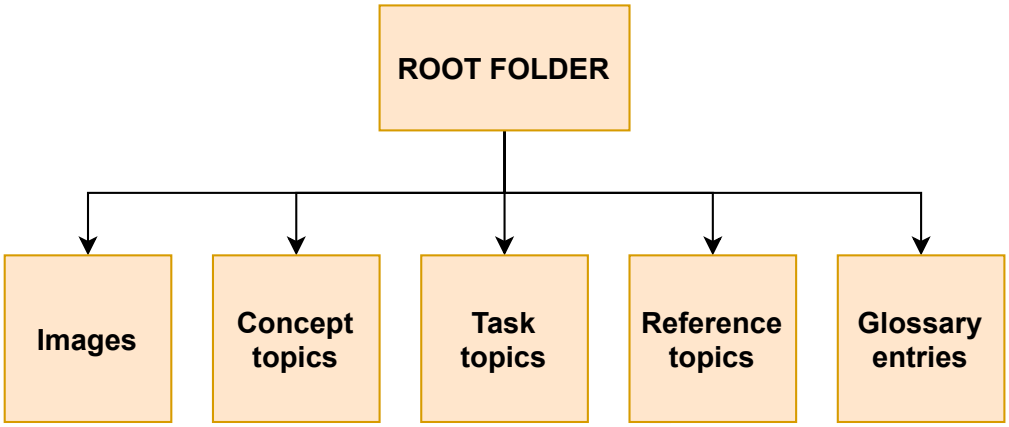
Prefixes vs. folders

It is common practice to prefix topic names with "c_", "t_", "r_" and "_g" for concepts, tasks, references, and glossary entries, respectively. For example:

- **c_magic_wand.dita**
- **t_how_to_cast_a_spell.dita**
- **r_wand_specifications.dita**
- **g_wand_shaft.dita**

For larger projects, it may be useful to create separate folders for topics of each type. This will make managing the project easier.

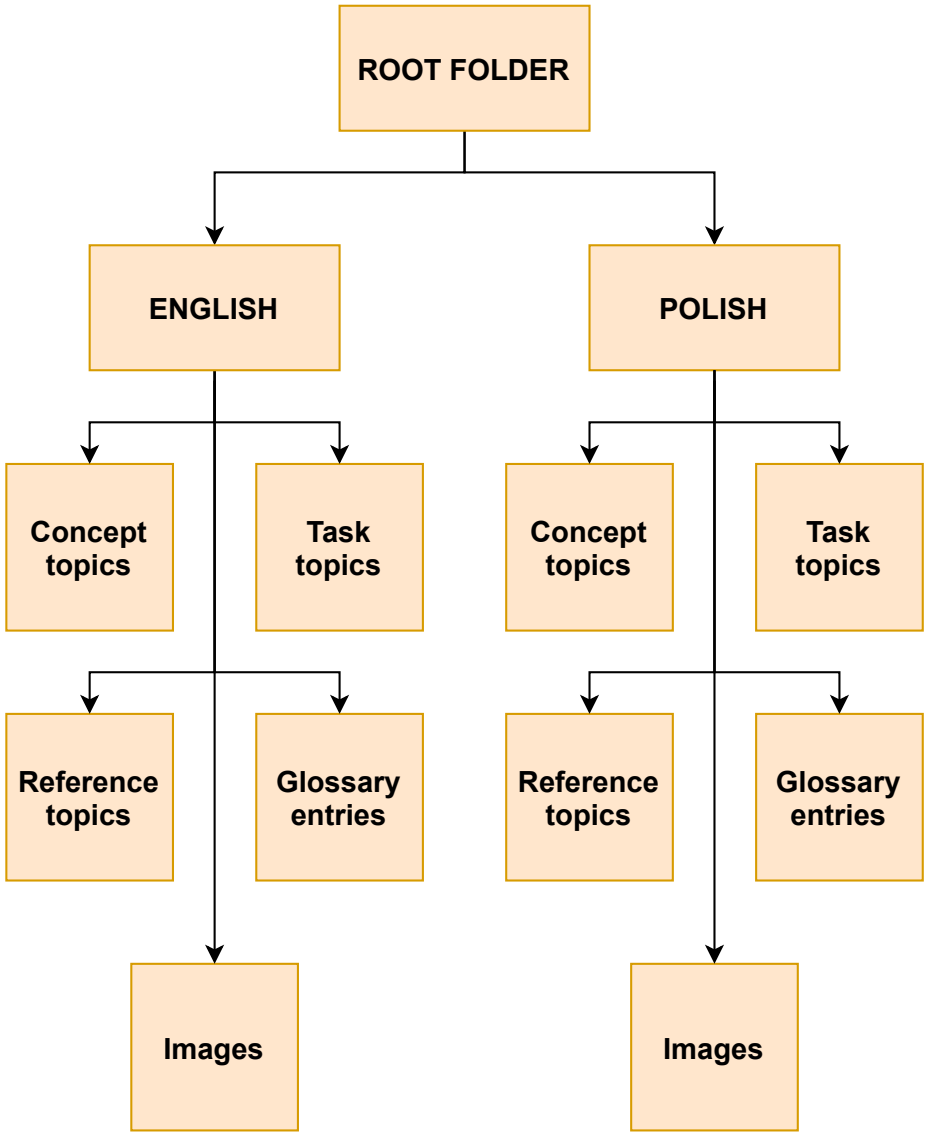
Figure 2. Example structure of a project with separate folders for different types of content



Language versions

To simplify localization, create a separate subfolder for each language version. Each subfolder should have the same structure.

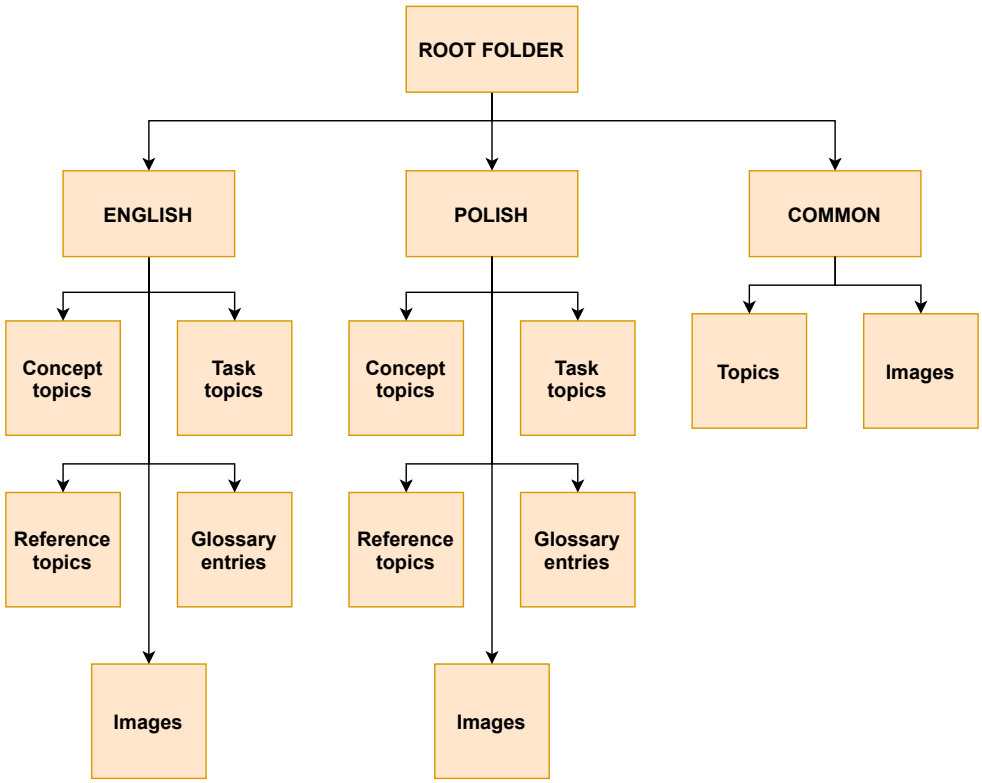
Figure 3. Example structure of a multilingual project with separate folders for different language versions



Files not to be translated

Some files, such as images and other content, are shared among language versions and don't need to be translated. It's a good idea to keep files you don't want to translate in a separate folder.

Figure 4. Example structure of a multilingual project with a separate folder for files you don't want to translate



Related information

Translation-friendly text *(on page 1)*

Content reuse *(on page 10)*

Importing a DITA project *(on page 15)*

Preparing DITA code for localization

Learn how to prepare your DITA code for localization using the `xml:lang` and `translate` attributes.

Setting the language of topics and maps

It's a best practice to specify the language of your DITA files, even if you're not planning localization. To specify the language and country of a topic or map, set the `xml:lang` attribute of its root element:

```
<concept id="c_magic" xml:lang="en-US">
```

Tip:
To set an attribute of an element in *oXygen XML Editor*, select the element, press **Alt + Enter** and type the name of the attribute you want to set.

Identifying content that you don't want to translate

To identify an element that you don't want to translate, set its `translate` attribute to "no":

```
<codeblock id="codeblock_spells" translate="no">
  let spellsLeft = 7;
</codeblock>
```

Related information

Non-translatable content *(on page 2)*

Text sorting and direction *(on page 9)*

Text sorting and direction

Learn about attributes and elements used to adjust text sorting and direction.

Sorting text using the sort-as element

In languages like English, you can easily sort lists by alphabetical order. But it's not that simple in languages such as Japanese and Chinese. This is because the same character can have different pronunciations depending on the meaning.

To specify text that you want to use for sorting, add a `sort-as` element inside a sorted element:

```
<glossentry id="gloss-harry-potter">
  <glossterm>&#x30CF;&#x30EA;&#x30FC;&#x30FB;&#x30DD;&#x30C3;&#x30BF;&#x30FC;</glossterm>
  <prolog>
    <sort-as>Harry Potter</sort-as>
  </prolog>
  <glossdef></glossdef>
</glossentry>
```

Sorting index terms using the index-sort-as element

You can adjust the sorting order of index terms by adding the `index-sort-as` element:

```
<indexterm>&#x30CF;&#x30EA;&#x30FC;&#x30FB;&#x30DD;&#x30C3;&#x30BF;&#x30FC;
  <index-sort-as>Harry Potter</index-sort-as>
</indexterm>
```

Changing text direction using the dir element

English and many other languages use left-to-right (LTR) script. But there are many languages like Hebrew or Arabic that use right-to-left (RTL) script. To optimize localization, it's a best practice to specify text direction using the `dir` attribute:

```
<map xml:lang="ar-IQ" dir="rtl">
```

Tip:
To set an attribute of an element in *oXygen XML Editor*, select the element, press **Alt + Enter** and type the name of the attribute you want to set.

Note:
You should apply a new language code and text direction *after* translation.

Related information

Preparing DITA code for localization *(on page 8)*

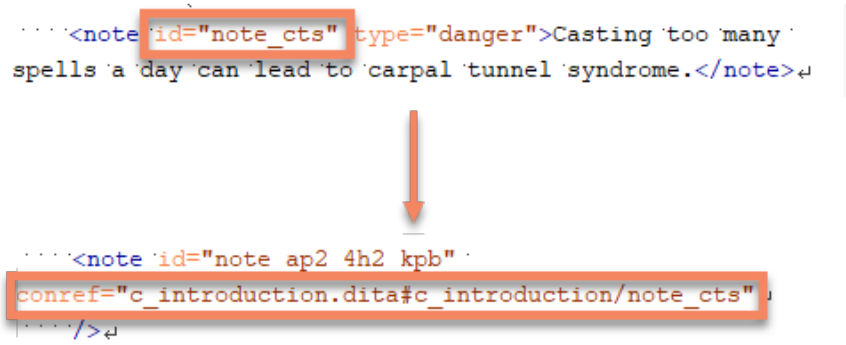
Content reuse

Learn best practices for content and keyword referencing to avoid problems with localization.

Handling conrefs

You can easily reuse content in DITA with the `conref` attribute.

Figure 5. Example of content referencing in DITA



Tip:
To set an attribute of an element in *oXygen XML Editor*, select the element, press **Alt + Enter** and type the name of the attribute you want to set.

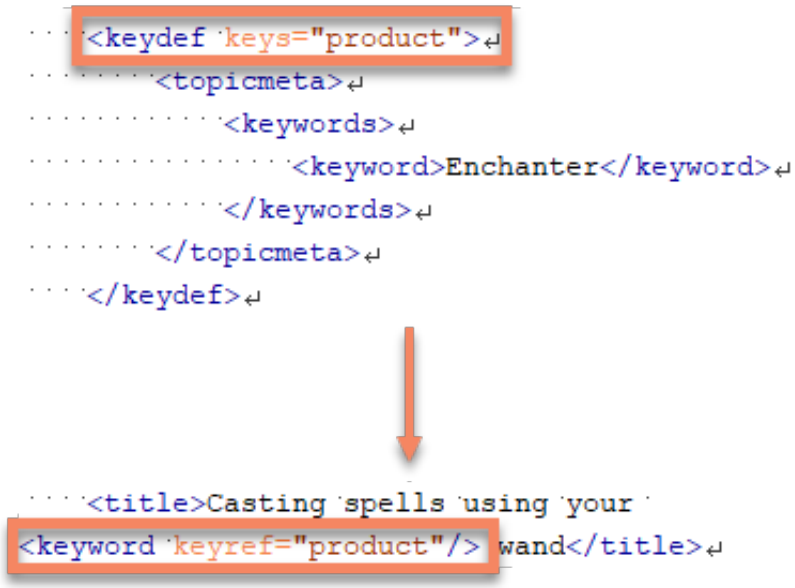
Improper use of content referencing can lead to translation errors (such as wrong grammatical forms or text not matching the context). To avoid such errors, follow these rules:

- Don't use conrefs for single words or parts of sentences. Instead, use them to reference whole sentences or larger fragments.
- Avoid excessive content reuse. This can make it harder for translators to know the context of the translated text.

Handling keyrefs

You can reference keywords using the `keyref` attribute. Keyrefs can be used for standard text or expressions that can change, such as product names.

Figure 6. Example of keyword referencing in DITA



When used improperly, keyrefs can lead to translation errors. You can minimize such errors by following these tips:

- Use keyrefs for terms that should not be modified in translation, such as product names or UI terms.
- Avoid using keyrefs for common nouns.
- Treat keyrefs as proper nouns and don't precede them with an article.

Tip:
To avoid translation errors, it may be a good idea to resolve all conrefs and keyrefs *(on page 12)* before localizing your project.

Related information

Resolving conrefs and keyrefs *(on page 12)*

Project structure *(on page 6)*

Resolving conrefs and keyrefs

Learn to quickly replace content references and keyword references with relevant content in oXygen XML Editor.

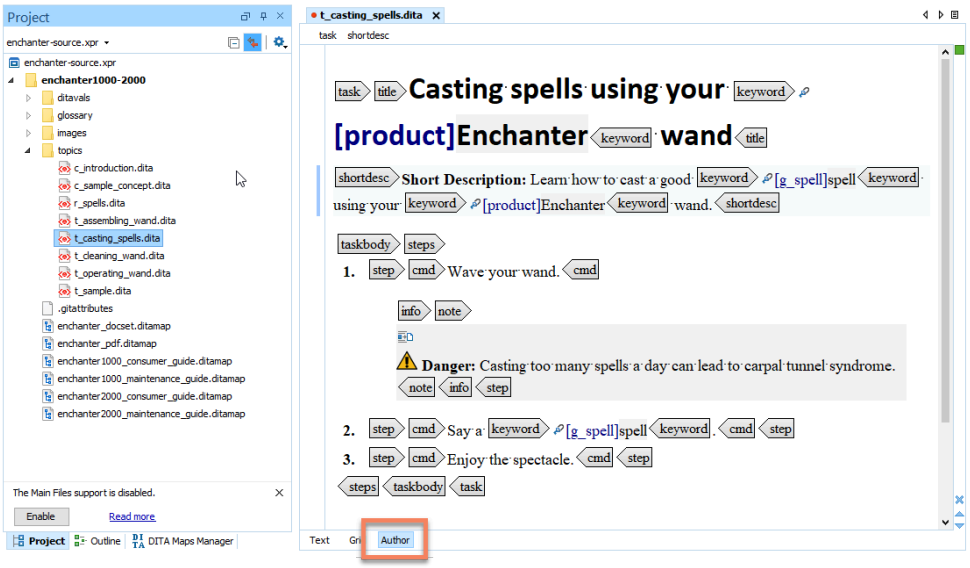
When preparing your project for localization, you can consider replacing content references (conrefs) and keyword references (keyrefs) with appropriate content.

Replacing conrefs and keyrefs with plain text may be useful, for example, when you have legacy content containing a lot of references and need to localize it without extensive rewriting.

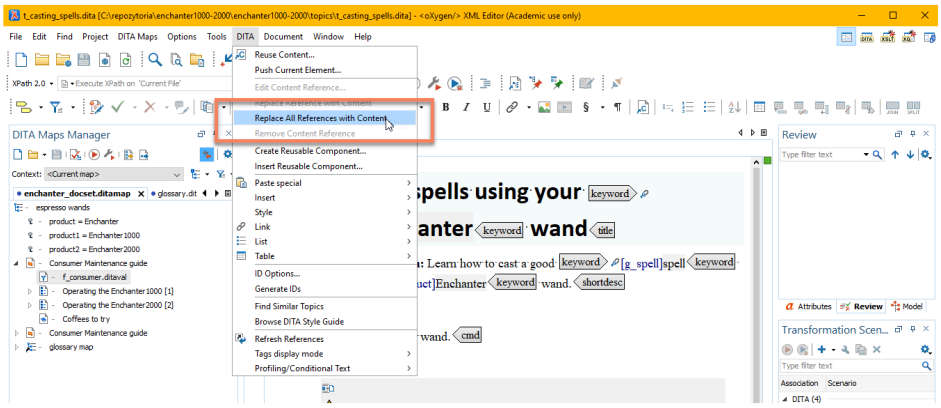
Important:
Once you resolve conrefs and keyrefs, they no longer exist in the localized text. You should decide what brings more benefits. If your project uses conrefs and keyrefs solely for non-translatable terms (such as trademarks), it may be useful to keep them. On the other hand, if such references point to single words or phrases, you may want to replace them with plain text to improve translation quality.

To resolve all conrefs and keyrefs in a topic in *oXygen XML Editor*, follow these steps:

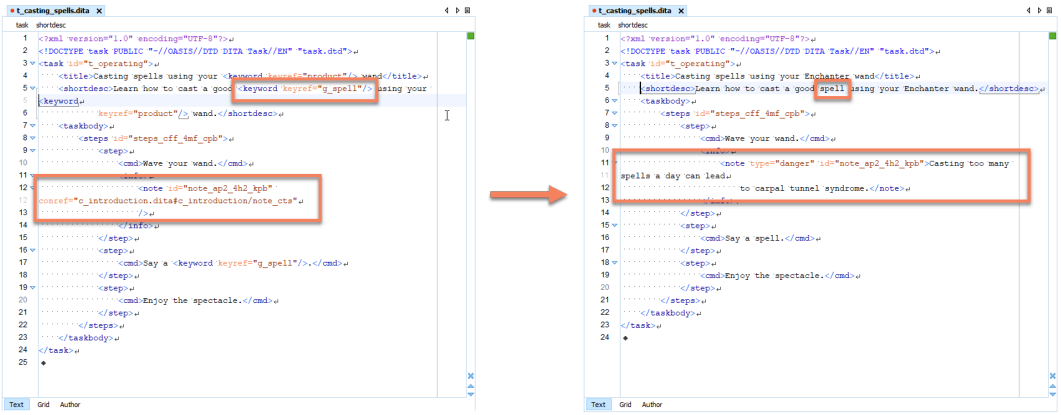
1. Open a topic containing references and switch to the **Author** mode.



2. Go to **DITA > Replace All References with Content**.



All references in the topic are replaced with relevant content.



Related information

Content reuse *(on page 10)*

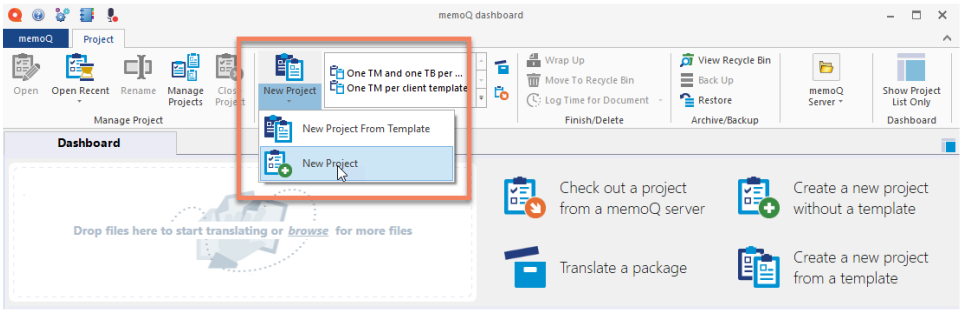
Chapter 3. Translating content in memoQ

Importing a DITA project

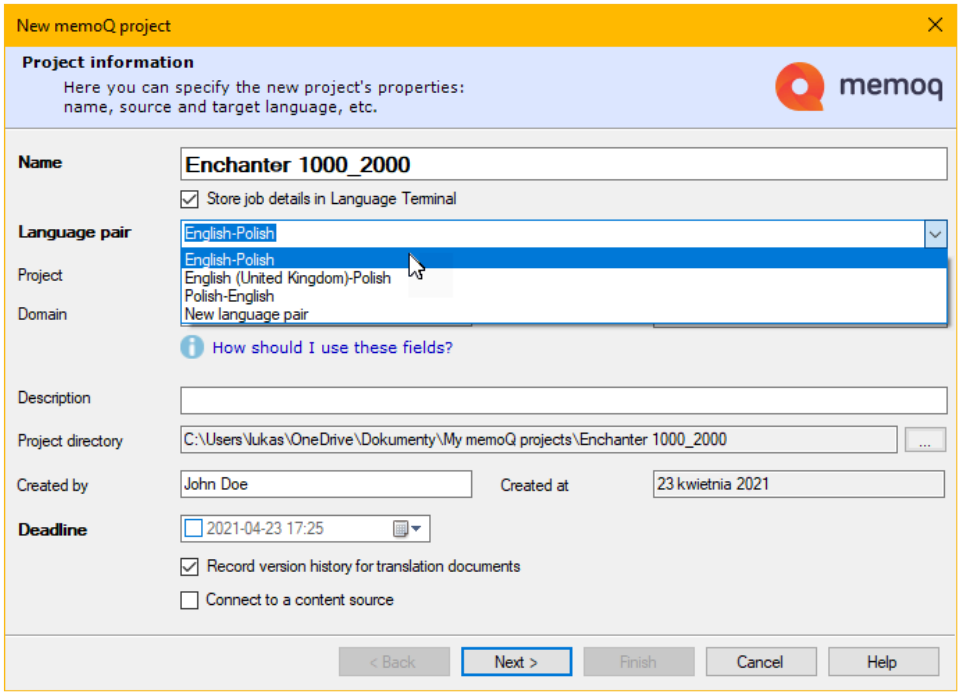
Learn how to import your DITA project into memoQ while preserving the project's folder structure.

When translating a DITA project, it's important to preserve the relationships between files. To import your project into *memoQ* without losing the project's folder structure, follow these steps:

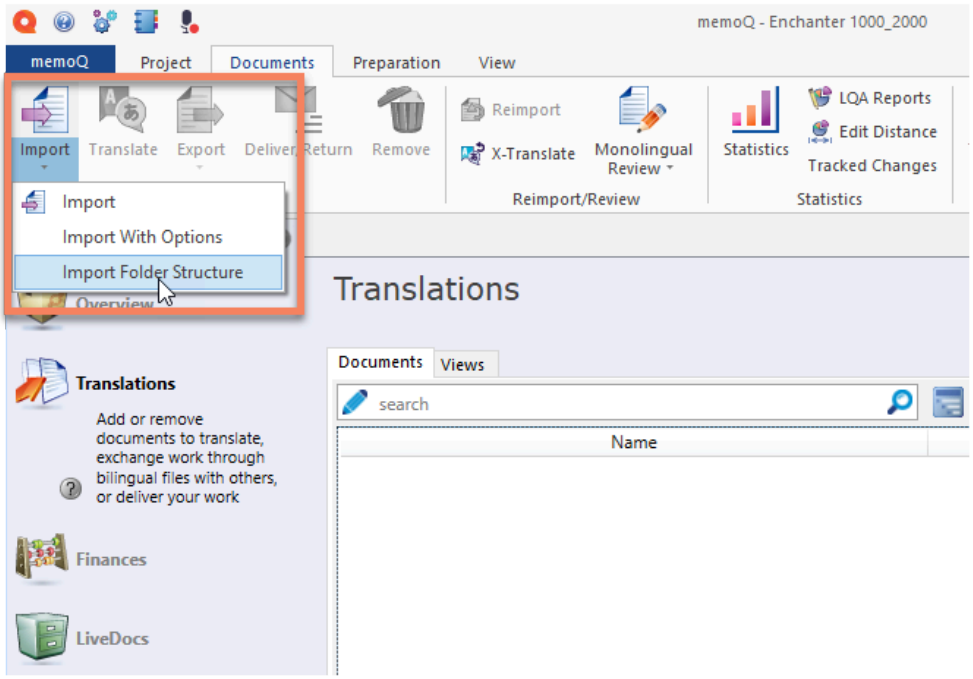
1. Go to **Project > New Project > New Project**.



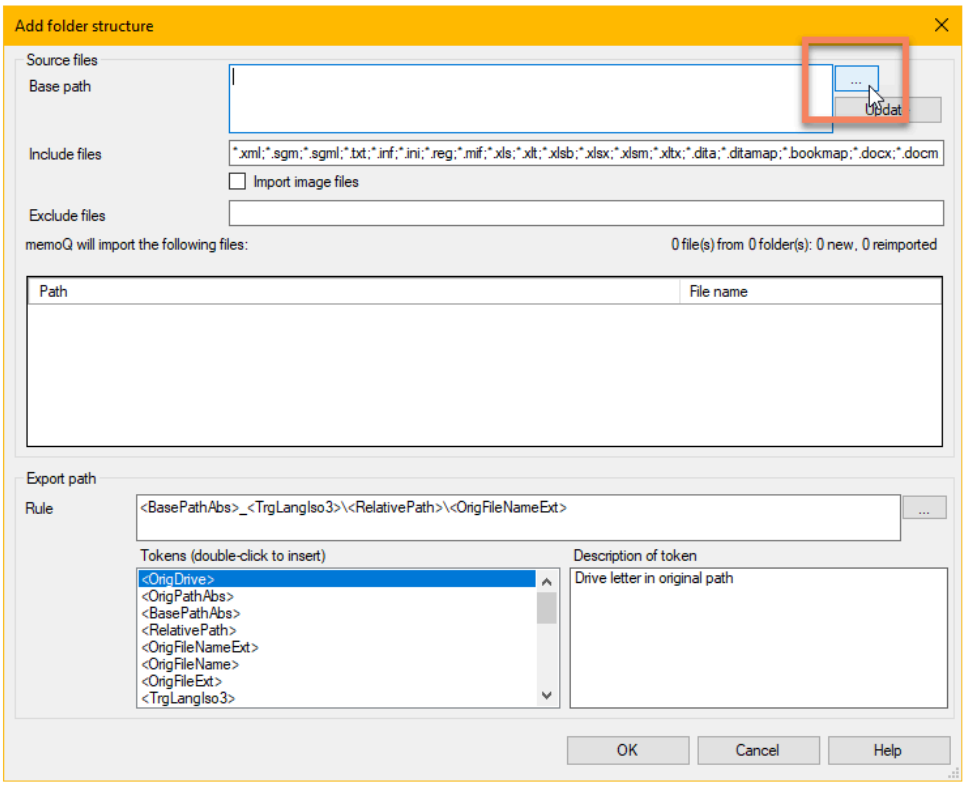
2. In the **New memoQ project** dialog, enter the **Name** of the project.
3. Choose a **Language pair**.



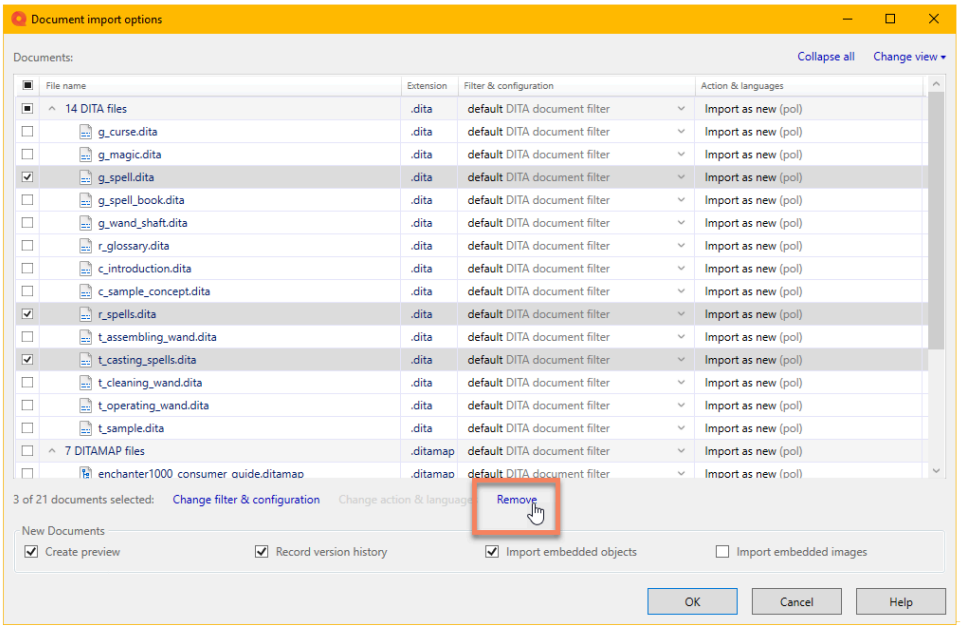
4. Enter the **Client** and any optional details, then click **Next**.
5. Skip the **Translation documents** step.
6. In the **Translation memories** step, choose or create translation memories.
7. In the **Term bases** step, choose or create term bases.
8. Click **Finish**.
9. Go to **Documents > Import > Import Folder Structure**.



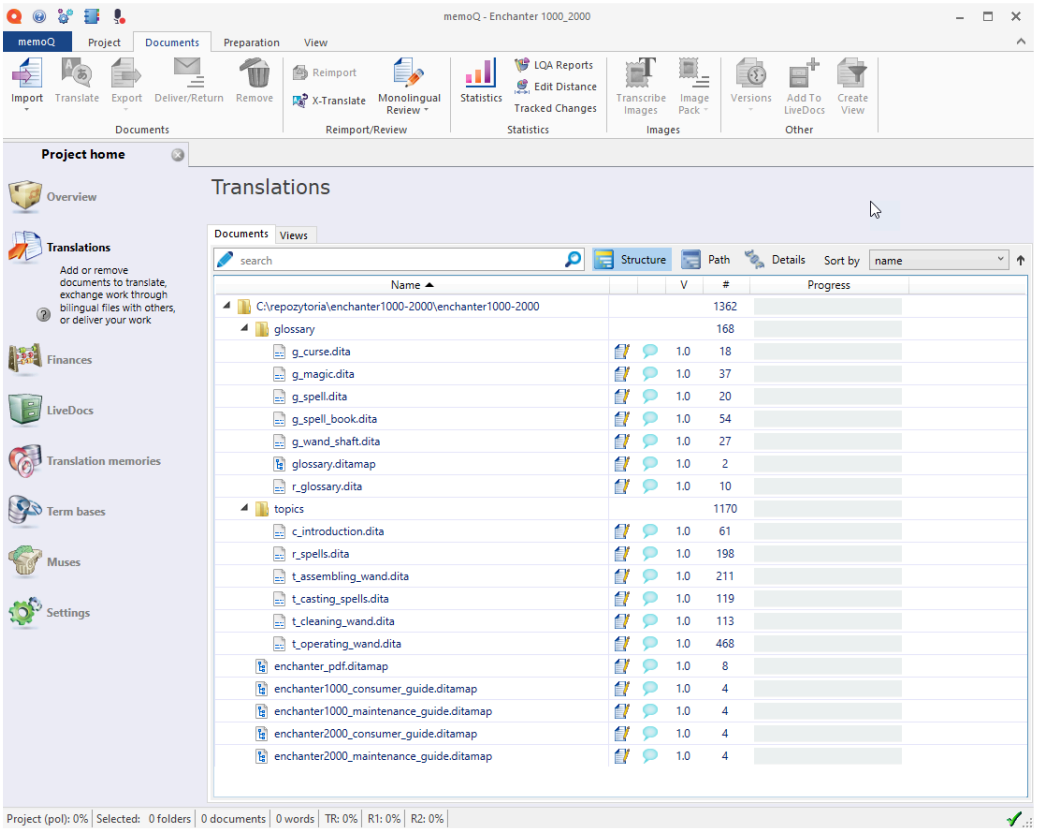
10. Click **…** next to the **Base path** field, choose the folder where you keep your project and click **OK**.



11. In the **Document import options** dialog, remove any files you don't want to translate and click **OK**.



Project files are ready for translation.



Tip:
To export a translated project, right-click on the root folder and choose **Export > Export (Stored Path)**.

Related information

Project structure *(on page 6)*

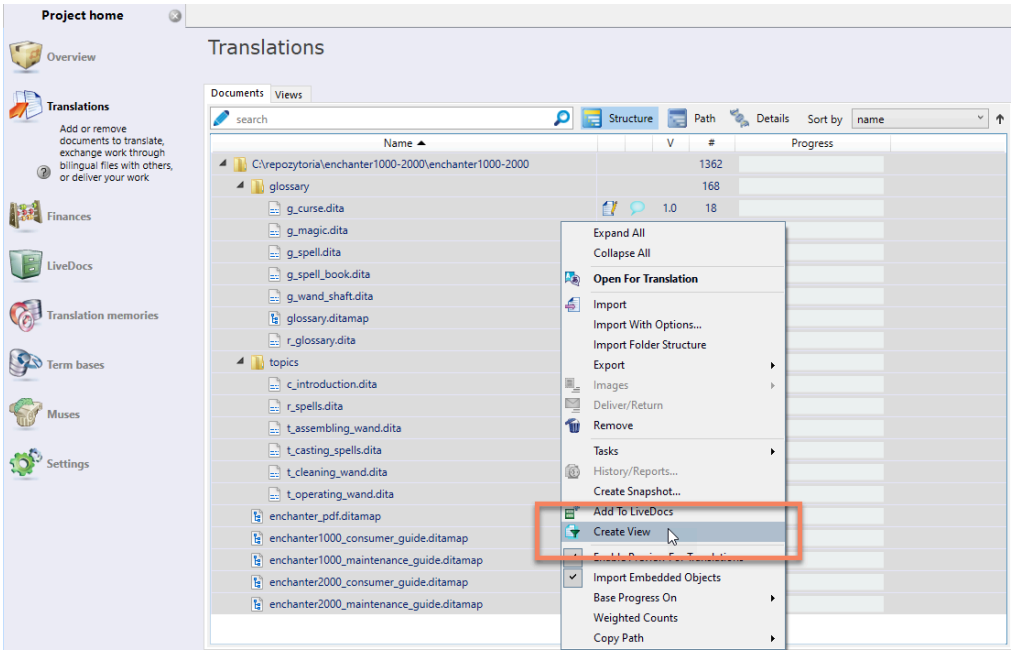
Translating repetitions *(on page 18)*

Translating repetitions

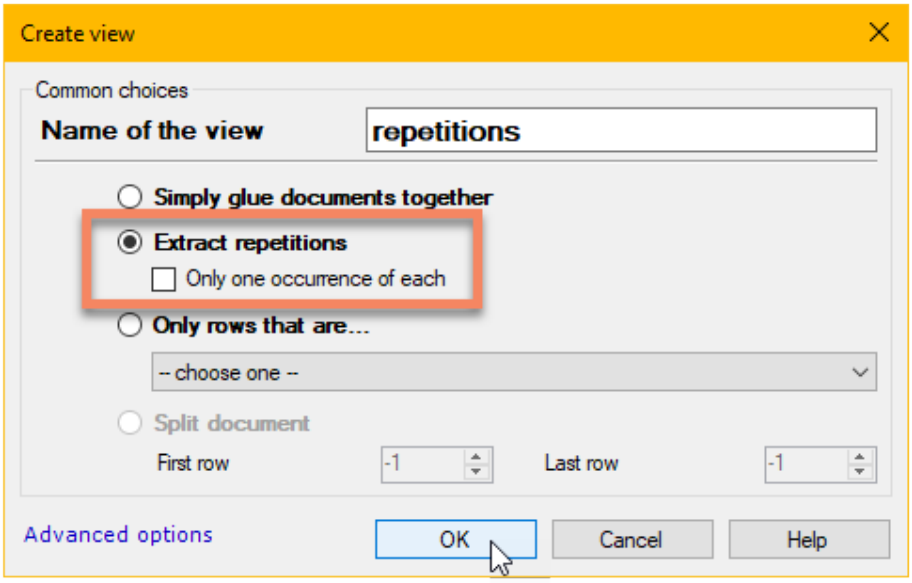
Learn how to simplify your translation process in memoQ by creating a view containing only repetitions.

If your DITA project contains a lot of repetitive content, it may be useful to include all repetitions in one view. To create a view containing all repetitions in *memoQ*, follow these steps:

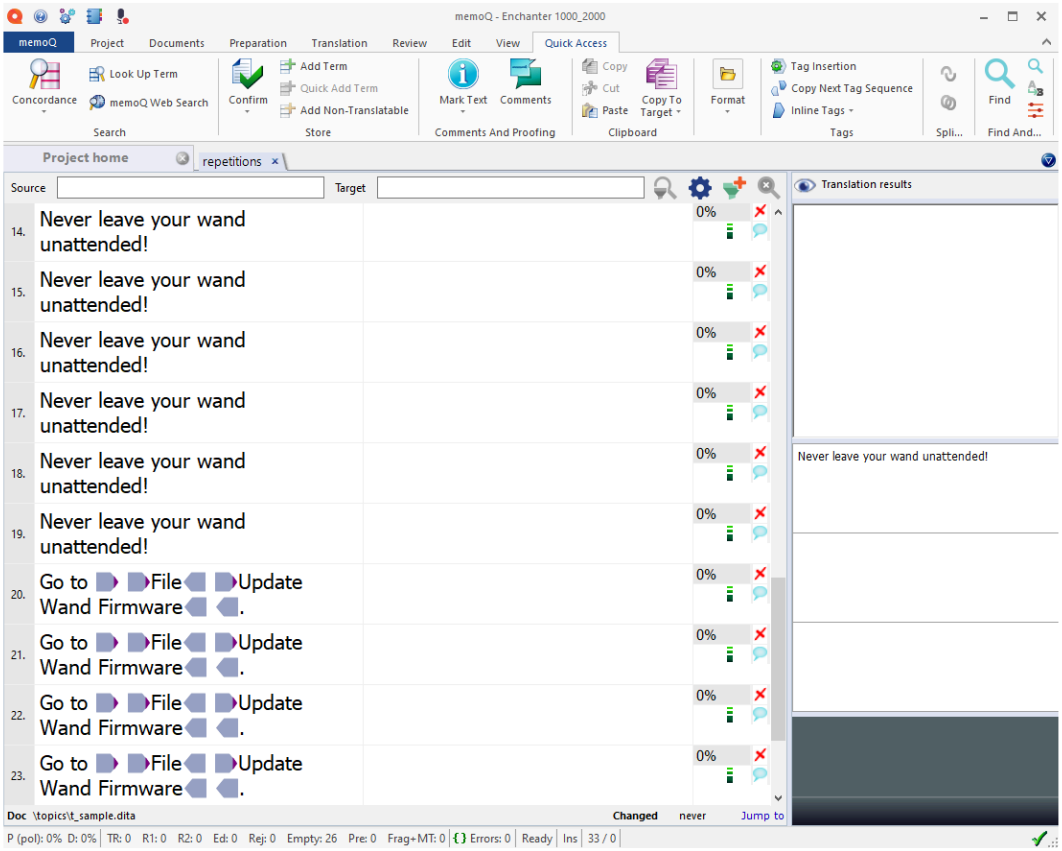
1. Go to **Project home > Translations** and select all files.
2. Right-click on the files and choose **Create View**.



3. In the **Create view** dialog, enter the **Name of the view**, select **Extract repetitions** and click **OK**.



Now you can work on a view containing all repeating segments in your project.



Note:

You don't always need to translate all repetitions the same way. Sometimes, the same text has different translations depending on the context.

Related information

Importing a DITA project *(on page 15)*

Chapter 4. Glossary

Learn basic terms associated with localizing DITA content.

CAT

Computer-Assisted Translation; the use of software tools (such as translation memory tools and terminology managers) to assist a human translator in the translation process.

CSV

Comma-Separated Values; a text file format that uses commas to separate values.

DITA

Darwin Information Typing Architecture; a standard XML-based architecture for creating topic-based documentation.

memoQ

A popular computer-assisted translation (CAT) software suite that runs on Microsoft Windows operating systems.

oXygen XML Editor

A popular XML editor offering advanced features for working with DITA content.

SVG

Scalable Vector Graphics; an XML-based vector graphics format.

XML

eXtensible Markup Language; a markup language designed to store and transport data in a format that is readable by both humans and machines.