TOC

How to create content for localization in MadCap Flare

TOC

Introduction

As companies expand globally, effective communication with international audiences becomes increasingly important. For technical communicators, this means ensuring that documentation is not only understandable but also culturally and linguistically appropriate for each target audience. This process goes far beyond simply translating words—it involves careful crafting and structuring of content to support seamless localization workflows.

This section introduces the fundamental concepts of translation and localization, and explains how MadCap Software's suite of tools—particularly MadCap Flare—supports technical writers and localization teams in managing multilingual content efficiently. Whether you're creating user manuals, help systems, or online documentation, understanding how to prepare your content for localization from the start is essential for maintaining consistency, reducing costs, and improving user experience across languages.

Translation and localization

Effective communication across languages and cultures is critical for global technical documentation. This is where translation and localization come into play. The terms translation and localization are often used interchangeably, but they refer to different stages of the internationalization process.

- Translation is the process of converting text from one language to another while preserving its meaning and intent. It focuses primarily on the linguistic aspect–making sure the content reads naturally and accurately in the target language.
- Localization, on the other hand, is a broader process that involves adapting the entire user experience for a specific market. This includes translating text, but also:
    - Formatting adjustments (for example: date/time, numbers, currencies)
    - Cultural adaptations (for example: imagery, symbols, and references)
    - UI alignment (for example: supporting right-to-left languages)
    - Legal or regulatory compliance for local audiences

Together, translation and localization ensure that users across the world can access content in a way that is intuitive, culturally appropriate, and fully functional.

Key challenges in translation and localization

Localizing technical content is not without challenges. Some of the most common include:

- Language structure and grammar: Languages vary in structure, grammar rules, and verbosity. Some languages (for example, German or Finnish) tend to expand significantly compared to English, affecting layout and UI design.
- Cultural sensitivities: Certain images, phrases, or symbols that are acceptable in one culture may be inappropriate or misunderstood in another.
- Formatting standards: Different countries use different conventions for dates, times, numbers, and currencies. These must be correctly adapted for each locale.
- Terminology consistency: Maintaining consistent use of product-specific terminology across all languages is critical for usability and professionalism.

- Technical constraints: Issues such as text expansion, character encoding (for example, UTF-8 vs. ASCII), or directionality (left-to-right vs. right-to-left) can introduce additional complexity.

Why localization matters in technical communication

Technical documentation is a crucial point of contact between a company and its users. Poorly translated or non-localized documentation can lead to confusion, frustration, and even product misuse. Effective localization:

- Improves user comprehension and satisfaction.
- Enhances the credibility of the product and the brand.
- Reduces support requests due to miscommunication.
- Ensures compliance with regional standards and regulations.

By planning for localization from the beginning of a project, technical communicators can significantly reduce effort, time, and cost in the long run. Localization should not be an afterthought. By integrating localization best practices into the writing and design process from the outset–such as writing in plain language, avoiding embedded text in images, and using structured content models–technical communicators can create documentation that is ready for global distribution with minimal rework.

MadCap Software products

MadCap Software offers a comprehensive suite of tools purpose-built for the creation, translation, and localization of professional technical documentation. These products are designed to work together seamlessly.

MadCap Flare

MadCap Flare is the flagship product of MadCap Software, designed for professional technical communicators who require powerful authoring and publishing capabilities. Flare stands out for its single-sourcing, topic-based architecture, and support for multiple output formats. From a localization perspective, it offers an extensive range of features that streamline the preparation of content for translation and simplify the ongoing management of multilingual projects.

Key features for localization:

- Topic-based authoring: Content is divided into modular topics rather than long monolithic documents. This allows for reusability and makes it easier to identify which pieces of content require translation when updates occur.
- Multilingual project architecture: Within a single Flare project, users can define language-specific targets, skins, and CSS styles. This avoids the need for duplicating entire projects for each language.
- Built-in terminology management: By using global variables for things like product names, UI elements, or legal disclaimers, consistency is maintained across the project, and translators can easily identify standardized terms.
- Conditional text and snippets: Flare allows writers to mark certain content as conditional, meaning it will only appear in certain outputs. Snippets can be reused across multiple topics, reducing translation costs and ensuring uniformity.
- Bi-directional and unicode support: Flare fully supports right-to-left (RTL) languages such as Arabic and Hebrew, as well as Unicode for global character compatibility.
- Review and translation tracking: Flare can identify which topics have been changed since the last translation export using the Analysis feature, making it easier to send only updated content to translators.
- Language skins and interface localization: Flare allows you to create language-specific HTML5 skins, adapting navigation elements and interface strings to match the language of the content.

MadCap Lingo

MadCap Lingo is a fully integrated Computer-Assisted Translation (CAT) tool designed to work directly with Flare projects. Unlike third-party CAT tools, Lingo understands the structure of Flare projects natively, eliminating the need to unzip, repackage, or reconstruct files for translation.

It is suitable for in-house translation teams as well as external localization vendors who need to work within a structured authoring framework.

Key features for localization:

- Seamless integration with Flare: Lingo can open Flare projects directly and maintain their structure, which reduces the chances of file corruption and translation errors. There is no need to flatten or simplify Flare content before translation.
- Translation memory (TM): Lingo automatically stores every translated segment in a database, so repeated phrases are translated consistently. TM also reduces costs over time, as content that has already been translated can be reused automatically.
- Terminology management: Users can create and manage term bases (glossaries) within Lingo. This ensures that translators use approved and standardized terminology, especially for technical terms, UI strings, or legal phrasing.
- Side-by-side editing interface: Lingo displays the source text alongside the translation field, allowing for easier editing and alignment. Users can preview formatting and tags in real time.
- Built-in QA checks: Lingo includes automated quality assurance features that identify common issues such as missing translations, broken links, tag mismatches, or inconsistent terminology usage.
- Project bundling and unpacking: Lingo allows users to create a translation bundle from a Flare project in XLIFF format, which can be shared with external translators who may or may not be using Lingo. Once translated, the bundle can be reimported seamlessly into Flare.
- Multi-language project management: Lingo supports the management of multiple target languages within the same project, enabling translation teams to work on several languages simultaneously and track progress.
- Reporting and metrics: Project managers can view reports on translation progress, TM leverage, QA issues, and term usage, helping maintain control over localization timelines and quality.

MadCap Capture

MadCap Capture is an advanced image capture and editing tool, critical for localizing visual content.

In multilingual documentation, visuals often need to be adapted. Capture helps by offering:

- Callouts: Use numbered or symbol-based callouts instead of embedded text.
- Single-source images: Use one image across different outputs with conditional text overlays.
- Annotations and layers: Keep image text layers editable for translation.
- Automated capture profiles: Consistent screenshots based on pre-set styles and regions.

Content strategy

In global technical communication, a well-defined content strategy is crucial. It's not just about creating content–it's about crafting content that's adaptable, reusable, and ready for localization. MadCap Flare offers a suite of tools that, when leveraged effectively, can streamline the creation and management of multilingual documentation. This section delves into the foundational elements of a robust content strategy: meticulous project planning, the principles of single sourcing, and strategic project structuring.

Project planning

Before initiating a Flare project with localization goals, thoughtful and structured planning is crucial. This stage sets the foundation for the overall project and influences every subsequent step—from content creation and translation to output and maintenance.

Define scope and objectives

Start by answering key strategic questions:

- What is the primary language, and which target languages are required?
- What deliverables are expected? (for example: HTML5 Help, PDFs, Knowledge Base articles)
- Who are the stakeholders, and what are their expectations?
- What platforms or CMS will host the final content?

These considerations help determine the scope of translation and localization, the tools you'll need, and the level of complexity involved.

Understand the audience

When planning for localization, it’s essential to recognize the cultural and contextual expectations of your audience. This includes:

- Regional language variants (for example, Brazilian Portuguese vs. European Portuguese)
- Reading direction (left-to-right vs. right-to-left)
- Technical literacy or tone preferences

This user-awareness will guide language choice, terminology, and even visuals.

Budget and timeline

Localization can be costly and time-consuming, especially when dealing with multiple outputs and languages. Early on, estimate:

- Cost per word for translation
- Time per translation cycle (authoring → translating → reviewing → publishing)

- Resources needed for internal QA and in-country review

Consider scheduling buffers for content changes late in the cycle or feedback from localization reviewers.

Select the right workflow and tools

Choosing the correct workflow is one of the most influential decisions in the planning phase. Will you be using:

- MadCap Lingo for integrated translation and project management?
- MadCap Flare output files?
- A third-party CAT tool, with XLIFF exports from Lingo?

These decisions will affect your folder structures, topic development, and conditional content management later in the process.

Establish translation criteria

Establish measurable criteria for what content should or should not be translated:

- Legal or region-specific information
- Embedded media
- UI elements and screenshots
- Content reused across products or versions

This ensures cost-effective localization and avoids wasting resources on unnecessary or low-impact content.

Documentation

Maintain clear documentation for:

- Project structure and naming conventions
- Translation requirements
- Contact points for translators and reviewers
- Review and sign-off procedures

Clear documentation improves consistency and speeds up onboarding for new contributors or translators.

Project structuring

Structuring your project well from the start saves time, reduces errors, and enables reusability of content across languages, outputs, and platforms.

Parent projects for shared content

If multiple projects share common content, consider creating a parent project that houses these shared elements. Other projects can then reference this parent project, ensuring consistency and simplifying updates across documentation sets.

Logical folder hierarchy

Structure your content in a way that reflects its logical flow and relationships. Group related topics into folders, and use descriptive naming conventions. This organization aids in navigation, both for authors and end users, and simplifies content management.

Language folders or projects

For multilingual content, consider how you'll structure your language versions:

- Single project with condition tags and language variables: More complex to manage, but efficient for shared content.
- Separate projects per language: Easier to isolate translations, but harder to maintain consistency.

Flare allows you to manage multiple targets from a single project, which can simplify maintenance when set up properly.

Standardize naming conventions

Consistent naming across topics, snippets, conditions, and variables helps streamline translation and avoids confusion. For example:

- Use prefixes like `t_`, `s_`, `img_` for topics, snippets, and images.
- Include language codes in translated versions (for example, `home_fr.htm` for French).

This standardization also helps when using scripts or translation tools to automate part of the workflow.

Use TOCs and targets thoughtfully

Define different Table of Contents (TOCs) and targets for each audience or language. This allows:

- Tailored navigation structures per language or product
- Conditional inclusion of topics
- Granular control over outputs (for example, disabling PDF output for certain languages)

Prepare for localization features

Use Flare's built-in features to enable localization from the outset:

- "Variables" on page 36: For branding, product names, legal text—translated once and reused across the project.
- "Snippets" on page 33: Ideal for content that repeats—a definition, warning, or disclaimer.
- "Conditions" on page 39: To manage content visibility per language or output.

Maintain documentation

Keep thorough documentation of your project's structure, conventions, and workflows. This practice is invaluable for onboarding new team members, ensuring continuity, and supporting localization efforts.

Single sourcing

In both content creation and localization, avoiding redundancy and maximizing efficiency are essential. One of the most powerful strategies to achieve this in MadCap Flare is single sourcing.

Single sourcing is the practice of writing and managing content once, and then reusing or repurposing it in multiple contexts, such as different outputs, audiences, or product versions, without duplicating the source material. This approach not only increases consistency and reduces maintenance effort, but it also provides a significant advantage during localization.

> Interestingly, single sourcing shares conceptual similarities with Translation Memory (TM) used in the localization process. TM is a database that stores previously translated segments (sentences, paragraphs, or phrases). When the same or similar content appears again, the translation system suggests the stored translation instead of translating it from scratch. In much the same way, single sourcing allows authors to reuse existing content blocks, reducing the number of segments that need to be created, reviewed, or translated.

By combining single sourcing strategies in Flare with the benefits of Translation Memory in tools like MadCap Lingo or third-party CAT tools, organizations can dramatically reduce translation costs, turnaround times, and the potential for inconsistency across languages and versions.

Why single sourcing matters for localization

When preparing content for translation, every unique segment adds to the cost and time involved. Repetitive, similar content must still be processed and reviewed unless a proper reuse strategy is in place. Without single sourcing, even small updates to shared content require manual tracking and retranslation. Single sourcing ensures that:

- You only write (and translate) content once.
- You can make global updates to content blocks, affecting every instance where they are used.
- Your localized versions stay consistent across different deliverables.

Key single sourcing features in Flare

Topic-based authoring

Flare's content model is based on individual topics rather than monolithic documents. Each topic represents a standalone piece of information, such as a procedure, concept, or reference, that can be assembled into multiple outputs

(like PDFs or online help). For localization:

- This ensures that only updated topics need to be sent for translation.
- Translators can work with smaller, more manageable content units.
- Topics can be reused across products, platforms, and documentation sets.

Snippets

Snippets are reusable chunks of content, such as recurring safety warnings, disclaimers, or standard instructions. Rather than copying and pasting the same content across topics, you can insert a snippet. This has multiple advantages:

- When the source snippet changes, every instance is updated automatically.
- Snippets reduce the volume of content that needs to be translated.
- Translators only need to translate the snippet once, regardless of how many times it appears.

Example use cases for snippets:

- Legal text blocks
- Common troubleshooting steps
- Headers and footers
- UI descriptions

Variables

Variables act as placeholders for text that may vary across outputs or languages, such as product names, company names, version numbers, or feature flags.

For example:

- `ProductName` = "Flare"
- `VersionNumber` = "2024.1"

Using variables:

- Ensures consistency across all content.
- Simplifies translation—translators don't need to worry about product or brand terms changing unexpectedly, or making a typo.
- Makes updates simple—you only change the variable value, not every occurrence in the content.

> Localization benefit: Only the variable definitions need to be translated once per language, not every occurrence in the document.

Conditional text

Conditional text allows you to include or exclude content based on defined conditions (like product version, region, or audience). This is particularly useful in multilingual or multi-product documentation scenarios and eliminates the need for separate projects for each variation, reducing duplication and improving scalability.

Target management

Targets in Flare define the output settings, such as the format (HTML5, PDF, Word), language, condition sets, variable definitions, and style sheets. Each target can generate a tailored version of your documentation, all from a single source.

Benefits summary

Benefit	Description
Consistency	Shared content, terminology, and UI references stay uniform across documents and languages.
Reduced translation costs	Translators only work on new or updated content; reusable content doesn't need to be retranslated.
Faster time to market	Localization can proceed faster when there's less unique content to process.
Simplified maintenance	Updates made to a single topic, snippet, or variable cascade across all outputs.
Scalability	Supports growing content libraries and multilingual expansions without an exponential increase in effort.

Best practices for effective single sourcing

- Plan reuse at the start: Identify shared content blocks, and design topics to be modular and independent.
- Avoid embedding content: Don’t write long, continuous narratives that are hard to separate. Break content into manageable, focused topics.
- Document your content strategy: Keep clear records of how variables, snippets, and conditions are used so that translators and future authors can understand the system.
- Test outputs frequently: Make sure that all targets (localized and otherwise) compile correctly with the right content conditions and variable definitions.
- Train your localization vendors: Ensure your translation team understands the use of snippets and variables so they don’t break reusable components during translation.

Writing for localization

Creating localizable content is a critical component in building user-friendly documentation that supports global audiences. While translation handles converting content from one language to another, localization is more holistic—it adjusts content to reflect cultural nuances, regulatory norms, and formatting expectations specific to each locale. This section outlines strategies and practices for writing content that is localization-ready from the outset, significantly improving quality and reducing translation costs and turnaround time.

General guidelines

The foundation of localization-friendly content starts with clarity, consistency, and simplicity. Source content should be easily understandable by translators and structured to avoid ambiguity, misinterpretation, or unnecessary complexity.

Key considerations:

- Write content assuming it will be translated into multiple languages.
- Keep terminology consistent across your documentation.
- Use tools like style guides, glossaries, and templates to standardize structure and tone.
- Plan for translation from the start, even if localization is not immediately required.

By focusing on high-quality source content, organizations set the stage for accurate and efficient localization later in the project lifecycle.

Language and sentence structure

Clear, simple language is the crucial for creating localization-friendly content. The way sentences are constructed—and the language used—directly affects not only how well content is understood by global audiences, but also how easily and accurately it can be translated, especially when using machine translation or translation memory systems.

Why simple language matters in localization

When source content is written clearly and simply:

- Translators can understand it faster and more accurately.
- Machine Translation (MT) tools produce better results.
- Localization costs are reduced due to fewer ambiguities or clarification needs.
- End users can quickly grasp the information, regardless of cultural or linguistic background.

> Writing in simple language is not "dumbing down" your content—it's about making your information clear, accessible, and translatable. It reduces misunderstandings, shortens localization timelines, and improves the user experience for global audiences.

Write in simple, global English

Use clear, straightforward vocabulary. Favor everyday terms over technical jargon or idiomatic expressions. Choose unambiguous language to reduce the risk of mistranslation or misunderstanding.

- Avoid idioms and metaphors

  ✕ *We're ironing out the details.*
  ✓ *We are finalizing the details.*

- Avoid slang and region-specific phrases

  ✕*Give it a shot.*
  ✓ *Try it.*

- Use standard English spelling and grammar

  Stick to consistent spelling conventions (for example, US English or UK English), and avoid switching within a project.

Use short, direct sentences

Sentence length and structure directly impact translatability. Shorter sentences are:

- Easier to understand
- Easier to parse by machine translation engines
- Easier to segment into translation memory systems

Best practices:

- Keep sentences under 20 words when possible.
- Break long, compound sentences into separate ideas.
- Eliminate unnecessary modifiers or qualifiers.

✗ *If the option is enabled and all the system requirements are met, the user will be able to continue using the software without further interruption.*
✓ *Enable the option. If system requirements are met, the software runs without interruption.*

Use the active voice

Active voice is clearer and more direct, which aids comprehension and makes it easier for translators to retain the meaning.

✗ *The system should be checked by the user.*
✓ *The user should check the system.*

This also helps clarify who does what, reducing ambiguity—especially in procedural or instructional content.

Use consistent terminology

Use the same term to refer to the same concept throughout your documentation. Inconsistent naming increases cognitive load for readers and creates extra work for translators, who may have to verify that two different terms refer to the same thing.

- Maintain a project-specific glossary.
- Define preferred terms in your style guide.

- Use MadCap Flare variables for product names, version numbers, or other repeated terms that may change between localizations.

Avoid ambiguity

Ambiguous language causes confusion in any language. When content is translated, ambiguity is amplified.

✕ *Click the button when it appears.*
✓ *Click the Install button when it appears in the toolbar.*

Be specific about:

- Objects (“this dialog” vs. “the confirmation dialog”)
- Actions
- Conditions and timing

Use parallel sentence structures

Use consistent syntax and structure for similar content to aid readability and translation memory reuse.

✓ Good:

- *Click the Save icon.*
- *Click the Print icon.*
- *Click the Export icon.*

✕ Inconsistent:

- *Click the Save icon.*
- *To print the document, select the Print icon.*
- *Export the file using the Export option.*

Prefer standard word order

Use Subject-Verb-Object (SVO) structure. This is the most common sentence format and is well-supported by MT tools.

✓ *The user enters a password.*
✕ *A password is required to be entered by the user.*

Consider the use of machine translation (MT) and translation memory (TM)

Simple, predictable sentence patterns benefit translation systems. Avoid creative or complex phrasings that might break matches in translation memory or reduce MT quality.

> Write like a machine will read it first—because it will.

Also avoid:

- Long noun chains: “user password expiration warning preferences”
- Pronoun ambiguity: “If it fails, restart it.” (What is “it”?)

Remember your audience

Writing for localization means writing for non-native speakers and users across cultures. Use inclusive and respectful language, and avoid assumptions about the user's technical level, background, or environment.

Error mitigation

Errors in localization projects often originate in the source content. These can propagate during translation and require costly post-editing and quality assurance.

Best practices to reduce errors in your writing:

- Check for linguistic accuracy: Typos, missing articles, or unclear phrasing may confuse both readers and translators.
- Minimize text in multimedia: Avoid embedding text in images or animations, which requires re-creation in each language.
- Handle placeholders and variables consistently: For example, always use consistent notation for product names or UI terms.
- Pre-validate content: Use MadCap Flare's built-in analysis tools to detect formatting issues, broken links, or duplicate IDs before export.

Style guides and glossaries

Standardizing how your team writes and translates terminology is essential for maintaining consistency, readability, and professional quality across all localized content. In multilingual projects, inconsistency can erode user trust, confuse readers, and increase translation costs due to avoidable rework. This is where style guides and glossaries become vital.

Why a style guide matters

A well-structured style guide serves as a reference document for authors, editors, and translators alike. It defines how content should be written and formatted in both the source and target languages. Key benefits include:

- Enforcing consistency: Ensures the same tone, sentence structure, and terminology across topics, contributors, and languages.
- Grammar and tone guidance: Clarifies whether content should be formal or conversational, active or passive, and provides examples of preferred constructions.
- Formatting standards: Specifies how to format:
    - Units of measurement (for example, metric vs. imperial)
    - Numbering and bullet styles
    - Capitalization rules
    - Quotation marks (for example, “English” vs. «French»)
    - Decimal separators (for example, 1,000.00 vs. 1.000,00)
- Audience and purpose alignment: Defines who the content is for (for example, technicians vs. end users), helping writers adjust complexity and tone accordingly.
- Language-specific notes: Style guides tailored to a specific language can include instructions on gendered language, formal address, and sentence length preferences.

Glossaries for terminology management

A glossary is more than a word list–it's a controlled vocabulary that shows approved translations of key terms. It’s especially important for:

- Technical terms: Ensures consistency in complex domains like software, engineering, or healthcare.
- UI strings and labels: Matches wording used in the software interface so users are not confused by mismatches between the UI and documentation.
- Untranslatable terms: Identifies brand names, product names, acronyms, or code elements that should remain in English or their source language.
- Avoiding false friends: Notes terms that shouldn't be translated literally to avoid miscommunication or cultural errors.

Benefits for the localization workflow

- Reduces translator queries and inconsistencies.
- Improves translation memory (TM) leverage.
- Ensures better QA outcomes with fewer revision cycles.
- Speeds up onboarding of new linguists or team members.

Accept changes

Before sending content for translation, it's critical to accept all tracked changes in your MadCap Flare project or source documents. While tracked changes are helpful during review, leaving them in place can cause serious issues in localization.

Why it matters

- Word count inflation: Both deleted and inserted text may be counted, increasing translation cost unnecessarily.
- Translator confusion: It's unclear which version of the text is final, leading to delays or errors.
- Tool compatibility: CAT tools (including MadCap Lingo) may not handle tracked changes correctly, which can cause formatting or segmentation issues.

Best practices

- Finalize edits before exporting: Ensure reviews are complete and content is stable.
- Accept or reject all changes: Use Flare's Review pane to confirm a clean state.
- Export only the finalized version: This ensures translators receive consistent, unambiguous input.

Communication with LSP

Effective communication with your Language Service Provider (LSP) is crucial for successful localization. Establishing clear, consistent, and proactive communication channels ensures that your content is accurately translated and localized, aligning with your project's goals and timelines.

Establish a collaborative relationship

- Early engagement: Involve your LSP during the planning stages of your project. Early collaboration allows for better alignment on project scope, timelines, and expectations.
- Understanding capabilities: Ensure that your LSP is familiar with MadCap Flare and its associated tools like MadCap Lingo and MadCap Capture, or find out which third-party tool will be used for localization of your project.

Provide comprehensive context

- Project overview: Share detailed information about your project's structure, objectives, and target audience. This includes providing access to style guides, glossaries, and any existing translation memories.
- Reference materials: Supply your LSP with relevant materials such as final PDFs, screenshots, and training guides. These resources offer context that aids in accurate translation.

Define clear processes and expectations

- Workflow alignment: Discuss and agree upon workflows, including file formats, delivery schedules, and quality assurance procedures. Clear workflows minimize misunderstandings and streamline the localization process.
- Feedback mechanisms: Establish channels for regular feedback and updates. Open communication allows for prompt resolution of issues and continuous improvement.

Leverage technology and tools

- Utilizing CAT tools: Whether using MadCap Lingo or other CAT tools like Trados Studio, memoQ, or Phrase (Memsource), ensure your LSP is proficient with the chosen platform. Compatibility with Flare exports (for example, XLIFF, HTML5, or PDF) is essential to maintain translation accuracy and efficiency.

- Translation memory (TM) and terminology management: Work with your LSP to build and maintain robust TMs and termbases. These resources promote consistency across translations and reduce costs over time.

Continuous collaboration and improvement

- Regular reviews: Conduct periodic reviews of translated content to ensure quality and adherence to guidelines. Collaborative reviews foster a sense of partnership and shared responsibility.
- Training and updates: Keep your LSP informed about updates to your products or services. Regular training sessions can help your LSP stay aligned with your evolving needs.

Prudent use of Flare features

MadCap Flare offers an extensive set of features for single-sourcing, modular content, and localization-ready authoring. However, careless or inconsistent use of these features can make projects harder to manage and significantly increase localization effort and cost. This section outlines how to effectively use some of Flare's most powerful features—snippets, variables, conditions, and links—and in a way that supports smooth and efficient localization.

Snippets

What they are:

Snippets in MadCap Flare are reusable blocks of content that can be inserted into multiple topics. They are like building blocks—if you create a chunk of content once (like a warning, a note, or a repeated procedure step), you can reuse it across your documentation by inserting it as a snippet.

A snippet behaves like a live reference: when you edit the snippet file, all instances of it update automatically wherever it has been inserted.

Why snippets matter for localization

When preparing content for translation, repetition equals cost. Repeating the same content across files can increase word count, translation time, and the risk of inconsistent translations. Snippets help eliminate that by:

- Reducing redundant content that needs to be translated.
- Ensuring consistent phrasing and structure across all localized outputs.
- Making updates easier, especially when changes occur late in the documentation cycle.

If a snippet is already translated once, most CAT tools will recognize it and reuse the existing translation, saving time and money.

Best practices for localization:

- Use for modular, repetitive content: Snippets reduce the number of words that require translation and keep content consistent. For example, a standard disclaimer used in 30 topics only needs to be translated once if reused via a snippet.
- Do not use in partial sentences: Translators do not see the whole sentence, each part is stored separately. Each snippet is its own file in a long list of other files, making it difficult for translators to know its context. Also, the content of the snippet may depend on the part of the sentence that is not snippetized, for example in inflected languages like Polish.
- Keep snippets self-contained: Avoid including full paragraphs or context-dependent content. Snippets should not depend on external content to be understood in translation.

- Avoid excessive nesting: Deeply nested snippets (snippets inside snippets) complicate both authoring and translation workflows. Limit to one level where possible.

✓ Good example:

```
<Snippet>Note: This feature is only available in the Pro version.</Snippet>
```

Why it's good:

- Reusable, clear, and context-independent.
- Easy to localize once and reuse globally.
- No embedded formatting or conditions.

✕ Poor example:

```
<Snippet>The error occurs if </Snippet> <Variable>[data element]</Variable> <Snippet>
is populated, but source is empty or null.</Snippet>
```

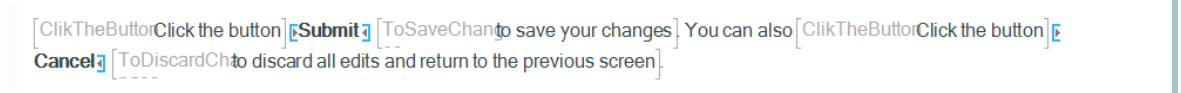
Problems:

- The translator sees both snippets as separate files. They probably appear in a way that the translator cannot recognize that they belong together.
- In some languages the grammatical structure of the order of the snippets may need to change.

> Tip: In Flare and Lingo, you can flatten the snippets and convert them to plain text. This way the translator sees the full contexts and can adjust the grammatical structure each time.

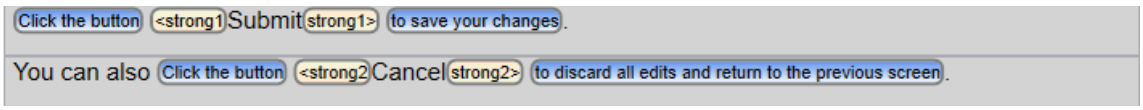
Translator's POV

Splitting a sentence into multiple snippets breaks sentence flow and removes necessary context for translators. This is how text with badly applied snippets may appear in MadCap Flare:



This is how a translator sees this text with the snippets maintained:

- MadCap Lingo



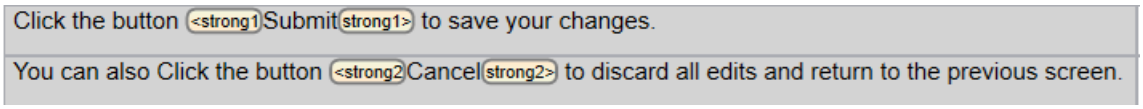
- SDL Trados Studio

Problems:

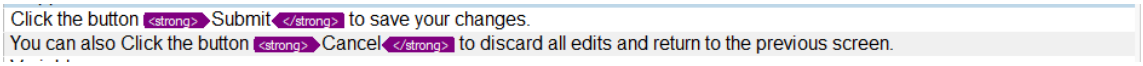
- The text is fragmented and lacks context.
- MadCap Lingo shows the content of a snippet; however it is not immediately editable as the snippet is stored in a separate file.
- In third-party CAT tools like SDL Trados Studio or memoQ, the translator does not immediately see the content of a snippet. They must locate it in the project files manually.
- Adjusting the grammatical structure of a sentence is nearly impossible due to the fixed nature of snippets.

MadCap Flare gives an opportunity to flatten snippets during the export process in order to remedy some of the possible problems. This is how the translator sees the text with snippets flattened:

- MadCap Lingo



- SDL Trados Studio



In the case of snippets, flattening them solves most of the problems. However, the snippet feature is lost in the translated files, and so are the benefits of using them in the first place.

Variables

What they are:

Variables are placeholders for content that may change over time or between outputs, such as product names, version numbers, or company names.

Best practices for localization:

- Use only for proper nouns, versions, and other non-translatable text: Product names, version numbers and branding should stay consistent and untranslated. Never use them for generic terms like "manual", "website", device".
- Avoid using variables mid-sentence for translatable words: Translators need full context, and segmented phrases with variables make proper sentence structure and grammar difficult in other languages.
- Use descriptive naming conventions: Use names like `ProductName_Pro`, not just `Var1`, to help translators and other authors understand the content.

✓ Good example:
The `<Variable>ProductName</Variable>` supports multiple file formats.

Why it's good:

- Clear, descriptive name
- Contains a proper noun that does not depend on the grammatical structure of the sentence.

✕ Poor example:
This `<Variable>EventType</Variable>` has already ended. You can start a new one.

- where `EventType` may be "webinar" or "meeting"

Problems:

In some languages, the variable may hold words that have different genders (in Polish, "webinar" is masculine (ten webinar) and "meeting" is neuter (to spotkanie)). This affects the whole sentence:

- *This `webinar` has already ended. You can start a new one.* becomes *Ten webinar już się zakończył. Możesz rozpocząć nowy.*

- *This `meeting` has already ended. You can start a new one.*becomes *To spotkanie już się zakończyło. Możesz rozpocząć nowe.*

> Tip: In Flare and Lingo, you can flatten the variables and convert them to plain text. This way the translator can adjust the grammatical structure each time.

Translator's POV

Using variables for common nouns makes it difficult—or even impossible—to account for varying genders, numbers, and grammatical cases in translated content. This is how text with badly applied variables may appear in MadCap Flare:

This DeviceType: printer is compatible with any PlatformNam operating system released after ReleaseDate June 2024.
Before using the DeviceType: printer, make sure your PlatformNam operating system is up to date.

This is how a translator sees this text with variables maintained:

- MadCap Lingo

This mc:v1 is compatible with any mc:v2 released after mc:v3.
Before using the mc:v4, make sure your mc:v5 is up to date.

- SDL Trados Studio

This <MadCap:variable name="Variables.DeviceType" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" /> is compatible with any <MadCap:variable name="Variables.PlatformName" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" /> released after <MadCap:variable name="Variables.ReleaseDate" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />.
Before using the <MadCap:variable name="Variables.DeviceType" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />, make sure your <MadCap:variable name="Variables.PlatformName" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" /> is up to date.

Problems:

- The translator does not see the content of a variable—neither in MadCap Lingo nor in a third-party CAT tool—as variables are stored in a separate file.
- Common nouns used as variables typically need to be translated and, in some languages, inflected or otherwise grammatically adjusted. This is not possible during translation due to the fixed nature of variables.

- Changing a variable may alter the gender of the noun it contains in the translated language. Such a change can require rewriting the surrounding sentence, which defeats the purpose of using variables to save time and effort.

MadCap Flare gives an opportunity to flatten variables during the export process in order to remedy some of the possible problems. This is how the translator sees the text with variables flattened:

- MadCap Lingo

This printer is compatible with any operating system released after June 2024.
Before using the printer, make sure your operating system is up to date.

- SDL Trados Studio

This printer is compatible with any operating system released after June 2024.
Before using the printer, make sure your operating system is up to date.

In the case of variables, flattening them solves most of the problems. However, the variables feature is lost in the translated files, and so are the benefits of using them in the first place.

Conditions

What they are:

Conditions in MadCap Flare are a content filtering mechanism that let you create multiple variations of your documentation from a single source. You apply condition tags to topics, paragraphs, or images to include or exclude content depending on the output target—for example, by language, audience, product version, or region.

This is essential for single-sourcing and supports scenarios such as:

- Showing different instructions for different software editions (for example, Standard vs. Pro)
- Hiding region-specific content from global outputs
- Outputting only content relevant to internal vs. external audiences

However, while conditions are helpful, they must be carefully managed when preparing content for localization.

Best practices for localization:

1. Limit the number of condition sets

Having too many condition tags or complex conditional logic can significantly complicate translation. Translators may receive content without knowing which combination of conditions will be visible together, increasing the risk of confusion or inconsistency.

2. Use clear, meaningful names

Avoid generic or cryptic names like `Cond1`, `Tag2`, or `OldNew`. Instead, use consistent, descriptive naming such as:

- `Language_FR`
- `Product_Pro`
- `Region_EU`
- `Audience_Admin`

This improves project maintainability and helps translators and reviewers better understand the context.

3. Avoid overlapping conditions on the same content block

Overlapping multiple condition tags on the same paragraph or sentence—especially when they are poorly documented—can lead to unpredictable output combinations and mistranslations. Instead, write out complete sentences of paragraphs and assign them the required conditions.

4. Document condition usage thoroughly

Create a conditions map or guide that explains:

- Which tags are used
- What content they affect
- Which outputs or locales they belong to

This is especially helpful for translation vendors working outside your authoring environment.

✓ Good example:

```
<p MadCap:conditions="Audience_Admin">Only administrators can access the system set-
tings via the Configuration panel.</p><p MadCap:conditions="Audience_EndUser">You can
change your personal preferences in the Settings menu.</p>
```

Why it's good:

- Conditions are applied at the paragraph level, keeping content separate and clear.
- The tags are descriptive and self-explanatory (`Audience_Admin`, `Audience_EndUser`), so there's no ambiguity.
- Translators can easily tell which content is intended for which user group, and you can generate audience-specific outputs without confusion.

✕ Poor example:

```
<p MadCap:conditions="UserType_Adv, Product_Old">This advanced feature is only
present in the old version.</p>
```

Problems:

- Multiple condition tags are stacked without clarity. It's unclear what the final output will include. If there's no documentation, a translator may not know which variant will be used or whether to translate both.

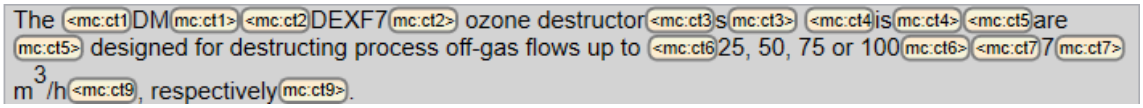
Translator's POV

Using conditions mid-sentence to create content variants can make it unclear what the final outputs will look like and harder for translators to work with. This is how text with badly applied conditions may appear in MadCap Flare:

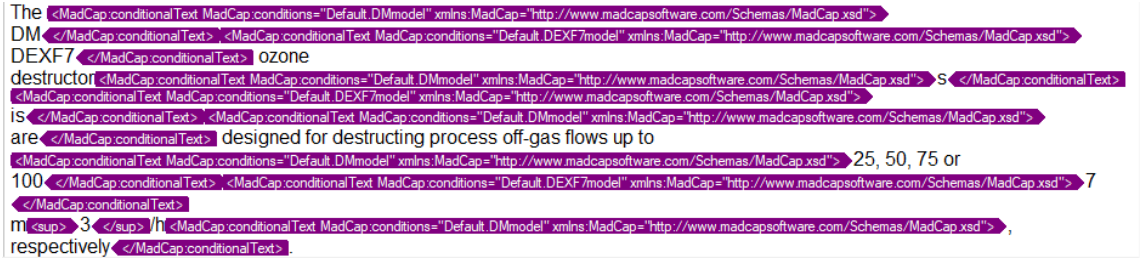
> The DM DEXF7 ozone destructor s is are designed for destructing process off-gas flows up to 25, 50, 75 or 100 7 m^3/h, respectively.

This is how a translator sees this text with conditions:

- MadCap Lingo



- SDL Trados Studio



Problems:

- An abundance of tags creates visual clutter, making it difficult to understand the sentence structure and anticipate the final outputs.
- Conditional text may require different grammatical structures in inflected languages, making accurate translation more difficult or even impossible.
- Creating plural forms of nouns by adding "-s" inside a conditional tag makes translation nearly impossible in many languages.

Hyperlinks vs. cross-references

What they are:

- Hyperlinks: Direct links to other topics or websites.
- Cross-references (xrefs): Dynamic links that include text from the target (like heading or title), and update automatically when the target changes.

Why it matters for localization

In multilingual projects, cross-references are usually preferable because they:

- Adapt to translated headings: Since xrefs pull the title from the target topic, the link text will automatically match the translated heading in the output.
- Reduce rework: If the title of a linked topic changes, you don't need to update every link manually.
- Minimize translation errors: Hyperlinks with custom link text can become outdated, mismatched, or inconsistent if not tracked carefully.

Best practices for localization:

- Use cross-references when linking to internal content: Always prefer cross-references when pointing to another topic or heading within the same project. This ensures the link remains valid even if the file name or heading changes, and the text will match the localized version.
- Avoid hardcoded text in hyperlinks: If you must use a hyperlink, avoid writing the link text manually. Instead, ensure it's clear and generic, or mark it for special attention in the translation process.
- Don't mix xrefs and hyperlinks inconsistently: Choose one method per context. For example, if you're consistently referencing other procedures or sections in a guide, stick with cross-references throughout to maintain a uniform structure.

✓ Good example:
For more information, see <xref href="installing.md"/>.
✕ Poor example:
For more information, click here.

The word "here" is not meaningful out of context and should be avoided in all documentation, especially translated content.

When hyperlinks are appropriate

There are some valid use cases for hyperlinks:

- Linking to external websites or external documents
- Email links (`mailto:`)
- URLs that will never change and don't require translation

In such cases, it's helpful to:

- Use variables for external URLs to centralize updates
- Use a consistent format for all such links

Tags placement

What it refers to:

How and where you place conditions, variables, or snippets within text can drastically affect translatability.

Best practices for localization:

- Apply tags at logical break points: Apply condition tags to entire paragraphs or block elements instead of splitting sentences. Inline tagging fragments the content and disrupts translation memory systems.
- Avoid tagging fragments of a sentence: Most translation tools work at the sentence level. If you apply condition tags to individual words, they may appear out of context to translators.
- Keep code clean: Avoid applying multiple types of tags (for example, snippet + variable + condition) within a short segment of text.

✓ Good example:

```
<p MadCap:conditions="Product_Pro">This feature is available in the Pro version only.</p>
```

Why it's good:

- Descriptive and precise name
- Applied to a single sentence for clarity
- No overlapping or compound conditions

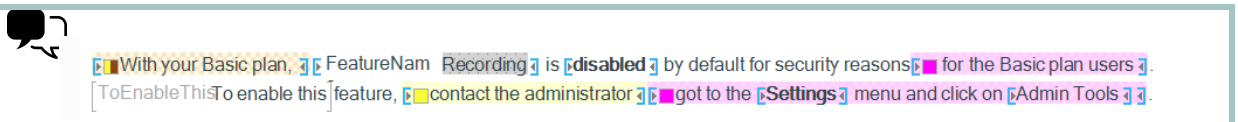
✕ Poor example:

```
<p>This feature is <span MadCap:conditions="Product_Pro">available</span> in the <Variable>ProductEdition</Variable>.</p>
```

Confusing to translate and maintain.

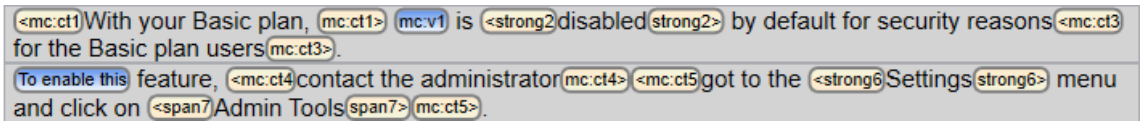
Translator's POV

Placing many different tags inside one sentence or overlapping tags can obscure the final output, make the content harder for translators to work with, and disrupt translation memory systems. This is how text with badly applied tags may appear in MadCap Flare:

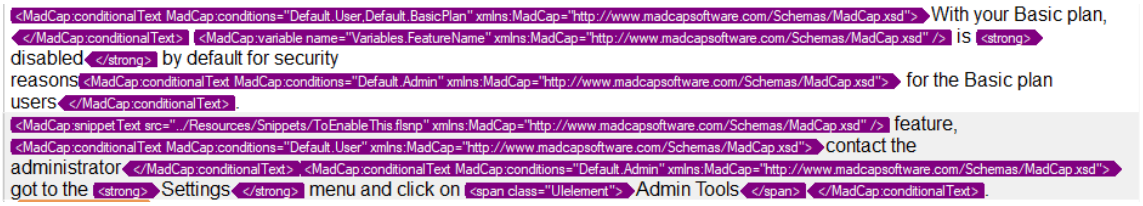


This is how a translator sees this text:

- MadCap Lingo



- SDL Trados Studio



Problems:

- An abundance of tags creates visual clutter, making it difficult to understand the sentence structure and anticipate the final outputs.

Flattening snippets and variables solves some potential issues, but not all–other tags may still remain and contribute to the confusion.

Formatting for localization

While writing content for localization, formatting is often overlooked as merely a visual concern. However, the way content is formatted—through stylesheets, page layouts, and UI structure—can significantly impact the cost, complexity, and quality of the localization process. Poor formatting decisions can result in hardcoded language, inaccessible styles, broken layouts in translated outputs, or even the need to localize CSS or redo print formatting per language.

This section focuses on specific formatting considerations that influence localization readiness, especially when working in a structured environment like MadCap Flare.

Stylesheets vs. inline formatting

What's the issue?

Inline formatting (like manually setting font colors, weights, or sizes in a WYSIWYG editor) may seem convenient, but it introduces major localization challenges.

✓ Best practices:

- Always define and apply styles via CSS classes or global stylesheets, not inline.
- Never use the `content` CSS property to inject visible or translatable text. Text embedded this way is hidden from translation workflows (like XLIFF exports).
- Keep separate stylesheets for languages only when absolutely necessary (for example, RTL layout support), but avoid language-specific styles for textual content.

✕ Pitfalls to avoid:

- Adding labels or titles via CSS, such as:

```
.warning::before { content: "Caution: "; }
```

This "Caution" will never be translated unless the stylesheet itself is extracted and localized manually.

- Applying formatting like `<span style="font-weight:bold; color:red;">Important</span>` inline. This is hard to manage at scale.

Better approach: Define `.important-note` in your CSS and apply it as a class:

```
.important-note { font-weight: bold; color: red; }
```

Then simply apply the class in Flare.

Allow for text expansion

Languages vary in length. When you translate English into German, French, or Russian, the resulting text may be up to 30-50% longer. Failing to allow space for this in your layout can lead to truncation, overflow, or broken page elements.

✓ Best practices:

- Design with flexible, responsive containers (not fixed-width text boxes).
- Leave extra space in UI mockups, tables, and callouts.
- Avoid narrow columns or tightly constrained layouts in PDF outputs.
- Consider previewing your content with "Lorem ipsum" text that simulates expansion (for example, with string duplication).

✕ Pitfalls to avoid:

- Hardcoded widths in tables or callouts that break when translated.
- UI screenshots that can't accommodate longer text labels.
- Vertical spacing that fits exactly around English but breaks in German or Finnish.

> **Example:** An English button labeled “Next” may become “Następny” in Polish – breaking UI layout if not designed to expand.

Paper size

When outputting to print (PDF, Word, and so on), paper size matters—both from a formatting and localization perspective.

✓ Best practices:

- Use A4 as your default paper size unless targeting North American audiences exclusively (where Letter is standard).
- If generating multilingual print outputs, create templates that adapt margins, spacing, and headers for both A4 and Letter formats.
- Be mindful of how expansion affects pagination and page breaks in different languages.

✕ Pitfalls to avoid:

- Locking your layout to one paper size without considering the locale.
- Setting hard-coded line spacing or paragraph lengths that cannot adjust dynamically with expansion.

> MadCap Flare lets you define multiple page layouts. Use these to tailor print output for different languages and regions.

Fonts

Font selection isn't just an aesthetic choice—it affects readability, character support, and cultural appropriateness in translated content.

✓ Best practices:

- Use Unicode-compliant fonts that support a broad range of characters (for example, Arial Unicode MS, Noto Sans, Segoe UI).
- Choose sans-serif fonts for online readability, and consider serif fonts for print if appropriate.
- Test that your font supports accented characters, Cyrillic, CJK characters (Chinese, Japanese, Korean), and right-to-left scripts.

✕ Pitfalls to avoid:

- Relying on custom or decorative fonts that lack character sets for other languages.
- Using fonts not installed by default in translation teams' environments (this may cause reflow or missing characters).
- Mixing font types within paragraphs in a way that breaks consistency post-translation.

> Google's Noto font family is a popular, localization-friendly font set designed specifically to handle many global scripts consistently.

Right-to-left (RTL) languages and directional icons

Right-to-left languages such as Arabic, Hebrew, Persian (Farsi), and Urdu pose specific formatting challenges—not only for text direction but also for visual cues, layout flow, and iconography.

RTL Considerations:

- Layout direction must be mirrored entirely—from paragraph alignment to navigation menus.
- Icons that suggest direction (for example, arrows, “Next” buttons, forward/back) must also be mirrored to align with reading flow.

✓ Best practices:

- Use Flare’s RTL support in stylesheets (`direction: rtl`) and page layouts.
- Maintain mirrored versions of icons when they indicate navigation or flow.
- Test the output visually in an RTL language to catch misalignments or broken elements.
- Ensure all style and condition tags are applied consistently across the mirrored layout.

✕ Pitfalls to avoid:

- Using “forward” arrow icons that point right → in Arabic UI (should point ← instead).
- Applying left-aligned styles to RTL content or forgetting to align headers and footers appropriately.
- Embedding text in images—RTL languages may require image mirroring, which becomes hard to do if text is baked in.

> **Example - directional icon mismatch:**
>
> A breadcrumb trail like “Home → Products → Details” should become “Details ← Products ← Home” in RTL output, along with matching icon direction.

Images and screenshots

Visual elements like images and screenshots are integral to technical documentation. However, without careful planning, they can become significant obstacles in the localization process. Challenges such as embedded text, fixed dimensions, and lack of scalability can lead to increased costs and time delays. This section outlines strategies to optimize images and screenshots for localization, leveraging tools like MadCap Capture and adhering to industry best practices.

Avoid embedded text

Embedding text directly into images (for example, using graphic design tools to place text within screenshots) renders the text inaccessible to translation tools. This practice requires manual extraction and re-embedding of translated text, increasing the risk of errors and inconsistencies.

Best practices:

- Use callouts and captions: Instead of embedding text, utilize callouts or captions that are separate from the image. MadCap Capture allows for the creation of callouts that are stored in an accompanying XML-based `.props` file, facilitating easy translation without altering the image itself.
- Leverage vector graphics: When possible, use vector graphics for illustrations. Vector formats enable the separation of text from images, allowing for straightforward text updates without modifying the graphic.
- Maintain text in source files: Keep all textual content within the source files of your documentation project. This approach ensures that translation memory systems can access and process the text efficiently.

> **Example:**
> ✕ Poor practice: A screenshot with embedded English text like "Click here to start" requires creating a new image for each language.
> ✓ Improved practice: A screenshot without text, accompanied by a caption or callout in the documentation that reads "Click the 'Start' button."

Allow for text expansion

Translated text often occupies more space than the original language. For instance, German translations can be up to 30% longer than their English counterparts. Without accommodating this expansion, text may overlap or extend beyond designated areas in images.

Best practices:

- Design flexible layouts: When creating callouts or annotations, ensure they can expand or contract based on text length. MadCap Capture supports auto-sizing of text boxes, which adjusts the size dynamically to fit the content .
- Test with pseudo-localization: Implement pseudo-localization techniques to simulate text expansion and identify potential layout issues before actual translation.
- Avoid fixed-size containers: Refrain from using fixed-size text containers in images. Instead, allow for dynamic resizing to accommodate varying text lengths.

> **Example:**
> ✕ Poor practice: A fixed-size callout box that fits only the English text "End" may not accommodate the Polish translation "Zakończ," leading to text overflow.
> ✓ Improved practice: A dynamically resizing callout box that adjusts to fit the translated text, maintaining readability and aesthetics.

Avoid hardcoded dimensions

Hardcoding image dimensions can lead to display issues across different devices and languages. Fixed sizes may not accommodate text expansion or varying screen resolutions, resulting in distorted or truncated images.

Best practices:

- Use relative sizing: Define image sizes using relative units (for example, percentages) instead of fixed pixels. This approach ensures images scale appropriately across different devices and layouts.
- Implement responsive design: Design images and layouts that adapt to various screen sizes and orientations, enhancing accessibility and user experience.
- Test across outputs: Preview images in different output formats (HTML5, PDF) to ensure consistent appearance and functionality.

> **Example:**
> ✕ Poor practice: An image set to a fixed width of 600 pixels may not display correctly on smaller screens or accommodate longer translated text.
> ✓ Improved practice: An image set to occupy 100% of the container width, allowing it to scale appropriately based on the viewing device.

Utilize layered files and MadCap Capture

Managing multiple versions of images for different languages can be time-consuming and error-prone, especially when dealing with embedded text and annotations.

Best practices:

- Adopt MadCap Capture: Use MadCap Capture to create layered images where text annotations are stored separately from the image in `.props` files. This separation simplifies the translation process and maintains consistency across languages.
- Maintain source files: Keep original, editable image files organized and accessible. This practice facilitates updates and modifications without recreating images from scratch.
- Integrate with translation workflows: Ensure that the text layers in images are included in the translation workflow, allowing translators to access and modify text without altering the image itself.

> **Example:**
> ✕ Poor practice: Creating separate images for each language, embedding translated text directly into each version.
> ✓ Improved practice: Using a single image with text layers managed through MadCap Capture, enabling efficient updates and consistency across all language versions.

Choose translation method

When preparing a Flare project for localization, it's important to choose the right translation method. Each method has specific advantages and disadvantages depending on your project size, complexity, timelines, and available resources. Here, we explore the most common approaches for translating MadCap Flare projects: using output files, using XLIFF, and using MadCap Lingo.

Quick comparison

Method	Best for	Main advantage	Main drawback
"Output files" on the next page	Small, simple projects	Easy for send out	Manual, error-prone translation and reintegration
"XLIFF" on page 60	Professional localization	Structure-preserving, efficient	Requires setup, more technical
"MadCap Lingo" on page 62	Integrated in-house workflow	Full Flare compatibility, QA built-in	Separate tool, licensing required

Output files

Seemingly one of the simplest methods for localization is generating output files (such as Word documents, HTML, or PDF) from Flare and sending these to the translator.

How it works

- You generate a finalized, human-readable output (for example, Word .docx).
- The translator edits the text directly in the output file.
- You reimport the translated document into MadCap Flare or manually update the project.

Pros

- Simplicity: Easy to understand for non-technical users and translators unfamiliar with Flare.
- Immediate usability: Translation can begin as soon as the outputs are ready without needing specialized translation tools.

Cons

- Manual rework: The translated file often has to be manually copied back into Flare topics or recreated, which is time-consuming and error-prone.
- Loss of structure: Outputs are not structured like the original Flare project. You lose snippets, variables, and metadata. You should avoid using conditions because they are not recognized and will not be applied correctly.
- Poor scalability: Suitable only for very small, static projects with minimal updates.

Best use cases

- Very small projects or single files without conditional text.
- One-time or occasional translation tasks with no planned updates.
- Non-technical audiences where translation needs to be fast and simple.

> It is recommended to always have the source files translated. By translating the source project (and not the output files), you increase the integrity of the project, you avoid potential workflow and maintenance challenges, and you reduce your overall translation costs.

XLIFF

XLIFF (XML Localization Interchange File Format) is an industry-standard file format specifically designed for exchanging translatable content between content creators and localization teams.

How it works

- You export content from Lingo as XLIFF files.
- The LSP or translator works directly with the XLIFF files using a Computer-Assisted Translation (CAT) tool.
- Once translation is complete, you import the XLIFF files back into Lingo, and then into Flare.

Pros

- Structured data: Maintains text structure, metadata, and segmentation, which makes reintegration much easier.
- Efficient for updates: Supports translation memory (TM), allowing reuse of previously translated segments.
- Better QA: CAT tools enable consistency checking, glossary matching, and error detection during translation.
- Automation-friendly: XLIFF workflows integrate well into automated localization pipelines.

Cons

- Export/import setup: You need to properly set up the export (manually select files, exclude non-translatable content, organize topics clearly).
- Error sensitivity: If not handled properly, structure mismatches or tag errors can break reimporting into Flare.

Best use cases

- Large or continuously updated projects.
- Projects needing ongoing translation memory management.
- Multilingual sites where accuracy and consistency are critical.

In MadCap Lingo, you can export XLIFFs for topics, snippets, and other assets. When translators return the completed XLIFF files, you can reimport them, and Flare automatically updates the content without breaking the project structure.

MadCap Lingo

MadCap Lingo is a translation management tool developed by MadCap Software specifically for working with Flare projects.

How it works

- You create a Lingo project from your Flare project.
- Lingo handles the extraction, translation, review, and reintegration of content.
- Supports TM, glossaries, machine translation integration, and quality assurance checks.

Pros

- Tight integration with Flare: Directly understands Flare structures–topics, snippets, variables, conditions, and even microcontent.
- All-in-one workflow: Project management, translation, and QA are all built into Lingo.
- Preserves structure: No risk of losing snippets, conditions, or complex linking.
- Facilitates in-house translation: Allows companies with internal translators to manage translation without the need for third-party CAT tools.
- Translation memory and glossary support: Automatically reuse previous translations across projects.

Cons

- Learning curve: Requires learning the Lingo interface and functionality.
- Licensing cost: Requires separate licensing from Flare.
- Limited LSP adoption: Not all LSPs are familiar with Lingo, so if you're outsourcing, you may need to export XLIFF instead.

Best use cases

- Organizations that want full control over the translation process.
- Projects with a lot of reuse (variables, snippets, conditions) that would be cumbersome to manage manually.

- Companies with in-house technical writers who also translate or manage localization.

> You can create a Lingo project from your Flare project. Once your in-house translator (or external linguist) completes the translation inside Lingo, you simply synchronize the translated content back into Flare with minimal manual effort.

Generate projects for translation

Before translation work can begin, your Flare project must be prepared and exported correctly. The method you choose will depend on the tools you and your localization partner (LSP) are using.

This section outlines three key procedures for exporting projects for translation:

- Exporting without MadCap Lingo (using output files)
- Exporting with MadCap Lingo (using XLIFF files)
- Exporting for direct translation in MadCap Lingo

Exporting output files

This method uses human-readable output files (for example, HTML or Word documents) exported from Flare. It is a basic approach when translators don't use CAT tools or when Lingo is not available.

1. Export your Flare project in the appropriate target format (Word or HTML) (to see this procedure, go to "Exporting Flare Project" on page 67 section).
2. Extract the files manually form the project's Output folder.
3. Provide the files to the LSP along with:
    - Any necessary instructions about formatting.
    - A style guide and glossary if available.
4. After translation:
    - The translator returns the translated Word or HTML files.
    - You manually copy and paste the translated content back into your Flare project.
    - Reapply snippets, variables, conditions as needed (if they were flattened in export).

Before you export with Lingo—preparation steps

To successfully export files for translation in MadCap Lingo or a third-party CAT tool, some setup is required. These preparation steps ensure that both your Flare project and the Lingo environment are correctly configured to ensure a smooth translation workflow.

Exporting Flare Project

You can export an entire Flare project, or parts of one, to use for translation. If you only need a portion of a parent project to be translated, you don't want to send the translator all of the files, but rather a smaller version of the project containing only the files requiring translation.

You can export a project using the Export Project Wizard to guide you. Alternatively, you can add an export file to your project and use the Export Project File Editor.

> NOTE: This procedure describes basic export configuration. For a comprehensive guide on exporting Flare Projects, visit the official MadCap Flare website: Exporting Projects

How to export a project via the Export Project Wizard

1. Open a project.
2. Select Project → Export Project. The Export Project Wizard opens.
3. In the New Project Name field, enter a name for the exported project.

   > (Optional) The New Project Path field is automatically populated with the default location (`Documents\My Project Exports`). If you want to export the project to a different location, click the browse ellipsis and select the folder you want.

4. In the Export From drop-down, select one of the options.
   - Entire Project: This option makes a copy of the entire project, including all folders, subfolders, and files.
   - Using Target: This option uses the same workflow as that used for generating a target. When you select a particular target, the same files and content that would be included in generated output are included in the exported project.
   - Using Conditions: This option exports only files and content affected by condition tags that you tell Flare to include or exclude.
   - Using File Tags: This option exports only files affected by file tags that you tell Flare to include or exclude. This method can be especially useful for translation purposes, exporting only files that are

marked with a certain file tag status (for example, Ready for Translation).

- Select Files: This option exports only specific files that you select. You can choose any files stored in the source project's Content Explorer and Project Organizer.

5. In the Output drop-down, select one of the options.

   - Zip File: This option packages the project files into a single zip file with an .flprjzip extension and places it in a location that you select. The default location is `Documents\My Project Exports`.
   - Project Files: This option simply exports the project files to a location that you select. The default location is `Documents\My Project Exports`.
   - Project Template: This option exports the project files to your templates folder (for example, `Documents\My Templates\Projects`). By being placed in this location, the project files become available as a template selection when you create a new project.

6. Click Next.

   (Optional) You can select one or both of the following:

   - Convert variables to text: Select this option if you want all relevant variables to be converted to text.
   - Convert snippets to text: Select this option if you want all relevant snippets to be converted to text.

7. Click Finish.

Creating Lingo Project

You can create a new Lingo translation project using the Start New Project Wizard. This wizard lets you select from many different kinds of files (for example, MadCap Flare projects, Doc-To-Help projects, Microsoft Word documents, Adobe FrameMaker, Adobe InDesign, XML files, DITA files), as well as specify one or more target languages. You can also add entire folders to a project and then select the file types you want to translate.

> NOTE: This procedure describes basic export configuration. For a comprehensive guide on exporting Flare Projects, visit the official MadCap Lingo website: Creating Projects

How to create a New Project

1. Do one of the following, depending on the part of the user interface you are using:

    - Ribbon: Select File → New Project.
    - Keyboard shortcut: Press CTRL+SHIFT+N.

    The Start New Project Wizard opens.

2. In the Name field, type an appropriate name for your project.
3. By default, a path to the `Documents\My Translated Projects` folder on your hard drive is entered in the Folder field. Continue with the next step, unless you want to have your project files placed in a different folder.

    > (Optional) In the Domain field, you can type a subject category for your project, or select one from the drop-down. Domain metadata appears throughout the project, such as when you select a suggestion in the translation memory.
    >
    > (Optional) In the Client field, you can type the name of the client you are creating a translation project for. Client metadata appears throughout the project, such as when you select a suggestion in the translation memory.

4. In the Source Language drop-down, select the original language that is used in the project you are translating. Make sure this language matches the source exactly. For example, you might select English (en) here, but the source files are actually in the English United States (en-us) language. Therefore, you should instead select English (United States) in this field.

5. In the Target Languages grid, select the language(s) that you want to use for the translation and click right arrow icon to move it to the Selected Languages grid. If you need to remove a language from the Selected Languages grid, click left arrow icon.

   (Optional) If you want to bind the Lingo project to a source control provider, select Bind to source control.

6. Click Next.

   (Optional) If you have selected the Bind to Source Control option, click Bind Project. The Bind Project dialog will appear. Complete all relevant steps and when you are finished, click Next.

7. Add files or folders to your Lingo project locally or from source control. If you select a folder, the files within it are added.

   To add files to a Lingo Project

   a. Click Add File.

   b. In the dialog that opens, navigate to the project or files that you want to translate. You may need to click the file type drop-down to view and select the appropriate kind of files you want to add.

   c. If you need to select only one file, you can double-click it. If you need to select multiple files (for example, multiple Word documents) in the same folder, you can hold your SHIFT or CTRL key and select either a range or individual files. Then click Open.

8. Click Next.

   (Optional) If you are creating a project using MadCap Flare or DITA source files, select additional project settings (flatten variables, flatten snippets, create language skin and so on).

9. Click Next.

   (Optional) Available translation memory (TM) databases are listed in a grid, and you can select the ones you want to use for the current project.

10. Click Next.

    (Optional) Available termbases are listed in a grid, and you can select the ones you want to use for the current project.

11. Click Finish.

When you create a new Lingo project, a file with an .liprj extension is automatically generated (for example, `My Project Polish.liprj`).

Exporting XLIFF files

XLIFF (XML Localization Interchange File Format) is a standard for exchanging localization data. Although Flare does not generate XLIFF files directly, you can export your Flare content to MadCap Lingo, and from there, create XLIFF files for translation.

1. Export your Flare project (to see this procedure, go to "Exporting Flare Project" on page 67 section).
2. Create a Lingo Project from Flare (to see this procedure, go to "Creating Lingo Project" on page 69 section).
3. Export an XLIFF bundle to send to your LSP:
    a. Select View → File List.
    b. Select the files to be included in the bundle.
    c. In the local toolbar click the Create Bundle button.
    d. In the Bundle Settings area, select the XLIFF Files Bundle and check the project files to include with the bundle.
    e. Include Project Translation Memories, as needed.
    f. Include Project Termbases, as needed.
    g. Click OK. Statistics for the project will appear.
    h. Click Close. A subfolder (within your project folder) named Exporting opens. This subfolder holds the ZIP file. This file contains the files that your translator needs to work with.
4. Deliver XLIFF files to the LSP along with a style guide, if available.
5. Once translation is completed and reviewed, import the XLIFF files back into the Lingo project.
6. Export the Lingo project into a Flare project in the target language.

Exporting for translation in Lingo

If your translation workflow is based inside your organization, or if your LSP uses MadCap Lingo, you can export and translate entirely within the Lingo environment–no need for separate files.

1. Export your Flare project (to see this procedure, go to "Exporting Flare Project" on page 67 section).
2. Create a Lingo Project from Flare (to see this procedure, go to "Creating Lingo Project" on page 69 section).
3. Translate directly in Lingo or export a LIPRJZIP bundle to send to your LSP. To create a LIPRJZIP bundle:
    a. Select View → File List.
    b. Select the files to be included in the bundle.
    c. In the local toolbar click the Create Bundle button.
    d. In the Bundle Settings area, select the Lingo Project Bundle and check the project files to include with the bundle.
    e. Include Project Translation Memories, as needed.
    f. Include Project Termbases, as needed.
    g. Click OK. Statistics for the project will appear.
    h. Click Close. A subfolder (within your project folder) named Exporting opens. This subfolder holds the LIPRJZIP file. This file contains the files that your translator needs to work with.
4. Send the LIPRJZIP file to the translator along with a style guide, if available.
5. Once translation is completed and reviewed, synchronize the translated content back into Flare.

Jak tworzyć treści do lokalizacji w MadCap Flare

Spis treści

Wprowadzenie

W miarę jak firmy rozwijają się na całym świecie, skuteczna komunikacja z międzynarodowymi odbiorcami staje się coraz ważniejsza. Dla osób zajmujących się komunikacją techniczną oznacza to zapewnienie, że dokumentacja jest nie tylko zrozumiała, ale także kulturowo i językowo odpowiednia dla każdej grupy docelowej. Proces ten wykracza daleko poza zwykłe tłumaczenie słów - obejmuje staranne tworzenie i strukturyzowanie treści w celu wsparcia płynnych przepływów pracy związanych z lokalizacją.

Ta sekcja wprowadza podstawowe pojęcia tłumaczenia i lokalizacji oraz wyjaśnia, w jaki sposób pakiet narzędzi MadCap Software - w szczególności MadCap Flare - wspiera autorów technicznych i zespoły lokalizacyjne w efektywnym zarządzaniu wielojęzyczną treścią. Niezależnie od tego, czy tworzysz podręczniki użytkownika, systemy pomocy czy dokumentację online, zrozumienie, jak przygotować zawartość do lokalizacji od samego początku, jest niezbędne do utrzymania spójności, zmniejszenia kosztów i poprawy komfortu użytkowania w różnych językach.

Tłumaczenie i lokalizacja

Skuteczna komunikacja między językami i kulturami ma kluczowe znaczenie dla globalnej dokumentacji technicznej. W tym miejscu do gry wkraczają tłumaczenie i lokalizacja. Terminy tłumaczenie i lokalizacja są często używane zamiennie, ale odnoszą się do różnych etapów procesu internacjonalizacji.

- Tłumaczenie to proces konwersji tekstu z jednego języka na inny przy jednoczesnym zachowaniu jego znaczenia i intencji. Skupia się przede wszystkim na aspekcie językowym - upewniając się, że treść jest czytana naturalnie i dokładnie w języku docelowym.
- Z drugiej strony, lokalizacja jest szerszym procesem, który obejmuje dostosowanie całego doświadczenia użytkownika do konkretnego rynku. Obejmuje to tłumaczenie tekstu, ale także
    - dostosowanie formatowania (na przykład: data/godzina, liczby, waluty)
    - adaptacje kulturowe (na przykład: obrazy, symbole i odniesienia)
    - Wyrównanie interfejsu użytkownika (na przykład: obsługa języków od prawej do lewej)
    - Zgodność prawna lub regulacyjna dla lokalnych odbiorców

Tłumaczenie i lokalizacja zapewniają użytkownikom na całym świecie dostęp do treści w sposób intuicyjny, odpowiedni kulturowo i w pełni funkcjonalny.

Kluczowe wyzwania związane z tłumaczeniem i lokalizacją

Lokalizacja treści technicznych nie jest pozbawiona wyzwań. Niektóre z najczęstszych to

- Struktura języka i gramatyka: Języki różnią się pod względem struktury, reguł gramatycznych i słownictwa. Niektóre języki (na przykład niemiecki lub fiński) mają tendencję do znacznego rozszerzania się w porównaniu do języka angielskiego, co wpływa na jakość tłumaczenia.
- Wrażliwość kulturowa: Niektóre obrazy, wyrażenia lub symbole, które są akceptowalne w jednej kulturze, mogą być nieodpowiednie lub źle rozumiane w innej.
- Standardy formatowania: Różne kraje stosują różne konwencje dotyczące dat, godzin, liczb i walut. Muszą one być prawidłowo dostosowane do każdej lokalizacji.
- Spójność terminologii: Utrzymanie spójnego stosowania terminologii specyficznej dla produktu we wszystkich językach ma kluczowe znaczenie dla użyteczności i profesjonalizmu.

- Ograniczenia techniczne: Kwestie takie jak rozszerzanie tekstu, kodowanie znaków (na przykład UTF-8 vs. ASCII) lub kierunkowość (od lewej do prawej vs. od prawej do lewej) mogą wprowadzać dodatkową złożoność.

Dlaczego lokalizacja ma znaczenie w komunikacji technicznej

Dokumentacja techniczna jest kluczowym punktem kontaktu między firmą a jej użytkownikami. Źle przetłumaczona lub niezlokalizowana dokumentacja może prowadzić do zamieszania, frustracji, a nawet niewłaściwego użytkowania produktu. Skuteczna lokalizacja:

- Poprawia zrozumienie i satysfakcję użytkowników.
- Zwiększa wiarygodność produktu i marki.
- Zmniejsza liczbę zgłoszeń do pomocy technicznej z powodu nieporozumień.
- Zapewnia zgodność z regionalnymi standardami i przepisami.

Planując lokalizację od początku projektu, komunikatorzy techniczni mogą znacznie zmniejszyć wysiłek, czas i koszty w dłuższej perspektywie. Lokalizacja nie powinna być kwestią drugorzędną. Włączając najlepsze praktyki lokalizacyjne do procesu pisania i projektowania od samego początku - takie jak pisanie prostym językiem, unikanie osadzania tekstu w obrazach i stosowanie ustrukturyzowanych modeli treści - komunikatorzy techniczni mogą tworzyć dokumentację, która jest gotowa do globalnej dystrybucji przy minimalnej ilości przeróbek.

Produkty MadCap Software

MadCap Software oferuje kompleksowy zestaw narzędzi przeznaczonych do tworzenia, tłumaczenia i lokalizacji profesjonalnej dokumentacji technicznej. Produkty te zostały zaprojektowane tak, aby płynnie ze sobą współpracowały.

MadCap Flare

MadCap Flare to flagowy produkt MadCap Software, przeznaczony dla profesjonalnych komunikatorów technicznych, którzy wymagają potężnych możliwości tworzenia i publikowania. Flare wyróżnia się architekturą opartą na pojedynczych źródłach i tematach oraz obsługą wielu formatów wyjściowych. Z perspektywy lokalizacji oferuje on szeroki zakres funkcji, które usprawniają przygotowanie treści do tłumaczenia i upraszczają bieżące zarządzanie wielojęzycznymi projektami.

Kluczowe funkcje lokalizacji:

- Tworzenie oparte na tematach: Treść jest podzielona na modułowe tematy zamiast długich monolitycznych dokumentów. Pozwala to na ponowne wykorzystanie i ułatwia identyfikację, które fragmenty treści wymagają tłumaczenia w przypadku aktualizacji.
- Wielojęzyczna architektura projektu: W ramach jednego projektu Flare użytkownicy mogą definiować specyficzne dla danego języka obiekty docelowe, skórki i style CSS. Pozwala to uniknąć powielania całych projektów dla każdego języka.
- Wbudowane zarządzanie terminologią: Dzięki zastosowaniu zmiennych globalnych dla takich elementów jak nazwy produktów, elementy interfejsu użytkownika lub zastrzeżenia prawne, spójność jest utrzymywana w całym projekcie, a tłumacze mogą łatwo zidentyfikować standardowe terminy.
- Tekst warunkowy i fragmenty: Flare pozwala pisarzom oznaczać określone treści jako warunkowe, co oznacza, że pojawią się one tylko w określonych wyjściach. Snippety mogą być ponownie wykorzystane w wielu tematach, zmniejszając koszty tłumaczenia i zapewniając jednolitość.
- Obsługa dwukierunkowa i Unicode: Flare w pełni obsługuje języki od prawej do lewej (RTL), takie jak arabski i hebrajski, a także Unicode dla globalnej kompatybilności znaków.
- Śledzenie recenzji i tłumaczeń: Flare może zidentyfikować, które tematy zostały zmienione od ostatniego eksportu tłumaczenia za pomocą funkcji analizy, co ułatwia wysyłanie tylko zaktualizowanych treści do tłumaczy.

- Skórki językowe i lokalizacja interfejsu: Flare umożliwia tworzenie skórek HTML5 specyficznych dla danego języka, dostosowując elementy nawigacyjne i ciągi interfejsu do języka treści.

MadCap Lingo

MadCap Lingo to w pełni zintegrowane narzędzie do tłumaczenia wspomaganego komputerowo (CAT), zaprojektowane do bezpośredniej współpracy z projektami Flare. W przeciwieństwie do narzędzi CAT innych firm, Lingo natywnie rozumie strukturę projektów Flare, eliminując potrzebę rozpakowywania, przepakowywania lub rekonstrukcji plików do tłumaczenia.

Jest to odpowiednie rozwiązanie zarówno dla wewnętrznych zespołów tłumaczeniowych, jak i zewnętrznych dostawców usług lokalizacyjnych, którzy muszą pracować w ustrukturyzowanych ramach autorskich.

Kluczowe funkcje lokalizacji:

- Płynna integracja z Flare: Lingo może bezpośrednio otwierać projekty Flare i utrzymywać ich strukturę, co zmniejsza ryzyko uszkodzenia plików i błędów w tłumaczeniu. Nie ma potrzeby spłaszczania lub upraszczania zawartości Flare przed tłumaczeniem.
- Pamięć tłumaczeń (TM): Lingo automatycznie przechowuje każdy przetłumaczony segment w bazie danych, dzięki czemu powtarzające się frazy są tłumaczone konsekwentnie. Pamięć tłumaczeniowa zmniejsza również koszty w czasie, ponieważ treści, które zostały już przetłumaczone, mogą być ponownie wykorzystane automatycznie.
- Zarządzanie terminologią: Użytkownicy mogą tworzyć i zarządzać bazami terminologicznymi (glosariuszami) w Lingo. Zapewnia to, że tłumacze używają zatwierdzonej i ustandaryzowanej terminologii, szczególnie w przypadku terminów technicznych, ciągów UI lub sformułowań prawnych.
- Interfejs edycji side-by-side: Lingo wyświetla tekst źródłowy obok pola tłumaczenia, umożliwiając łatwiejszą edycję i wyrównanie. Użytkownicy mogą podglądać formatowanie i tagi w czasie rzeczywistym.
- Wbudowane kontrole jakości: Lingo zawiera zautomatyzowane funkcje zapewniania jakości, które identyfikują typowe problemy, takie jak brakujące tłumaczenia, niedziałające linki, niedopasowanie tagów lub niespójne użycie terminologii.
- Łączenie i rozpakowywanie projektów: Lingo umożliwia użytkownikom tworzenie pakietów tłumaczeń z projektu Flare w formacie XLIFF, które mogą być udostępniane zewnętrznym tłumaczom, którzy mogą lub nie mogą korzystać z Lingo. Po przetłumaczeniu pakiet może zostać ponownie zaimportowany do Flare.
- Zarządzanie projektami w wielu językach: Lingo obsługuje zarządzanie wieloma językami docelowymi w ramach tego samego projektu, umożliwiając zespołom tłumaczeniowym jednoczesną pracę nad kilkoma

językami i śledzenie postępów.

- Raportowanie i metryki: Kierownicy projektów mogą przeglądać raporty dotyczące postępów w tłumaczeniu, dźwigni TM, kwestii kontroli jakości i użycia terminów, pomagając zachować kontrolę nad terminami lokalizacji i jakością.

MadCap Capture

MadCap Capture to zaawansowane narzędzie do przechwytywania i edycji obrazów, niezbędne do lokalizacji treści wizualnych.

W wielojęzycznej dokumentacji często zachodzi potrzeba dostosowania wizualizacji. Capture pomaga oferując:

- Objaśnienia: Używanie numerowanych lub opartych na symbolach objaśnień zamiast osadzonego tekstu.
- Obrazy z jednego źródła: Używanie jednego obrazu w różnych materiałach wyjściowych z warunkowymi nakładkami tekstowymi.
- Adnotacje i warstwy: Warstwy tekstowe obrazu mogą być edytowane na potrzeby tłumaczenia.
- Zautomatyzowane profile przechwytywania: Spójne zrzuty ekranu oparte na wstępnie ustawionych stylach i regionach.

Strategia treści

W globalnej komunikacji technicznej dobrze zdefiniowana strategia treści ma kluczowe znaczenie. Nie chodzi tylko o tworzenie treści - chodzi o tworzenie treści, które można dostosować, ponownie wykorzystać i przygotować do lokalizacji. MadCap Flare oferuje zestaw narzędzi, które przy efektywnym wykorzystaniu mogą usprawnić tworzenie i zarządzanie wielojęzyczną dokumentacją. W tej sekcji omówiono podstawowe elementy solidnej strategii treści: skrupulatne planowanie projektu, zasady pojedynczego pozyskiwania i strategiczne strukturyzowanie projektu.

Planowanie projektu

Przed rozpoczęciem projektu Flare z celami lokalizacyjnymi kluczowe jest przemyślane i ustrukturyzowane planowanie. Etap ten stanowi podstawę dla całego projektu i wpływa na każdy kolejny krok - od tworzenia treści i tłumaczenia po produkcję i konserwację.

Określenie zakresu i celów

Zacznij od odpowiedzi na kluczowe pytania strategiczne:

- Jaki jest język podstawowy i jakie języki docelowe są wymagane?
- Jakie rezultaty są oczekiwane? (na przykład: Pomoc HTML5, pliki PDF, artykuły bazy wiedzy)
- Kim są interesariusze i jakie są ich oczekiwania?
- Jakie platformy lub CMS będą hostować ostateczną zawartość?

Rozważania te pomagają określić zakres tłumaczenia i lokalizacji, potrzebne narzędzia i poziom złożoności.

Zrozumienie odbiorców

Podczas planowania lokalizacji niezbędne jest rozpoznanie kulturowych i kontekstowych oczekiwań odbiorców. Obejmuje to:

- Regionalne warianty językowe (na przykład portugalski brazylijski vs. portugalski europejski)
- Kierunek czytania (od lewej do prawej lub od prawej do lewej)
- Umiejętności techniczne lub preferencje dotyczące tonu

Ta świadomość użytkownika będzie kierować wyborem języka, terminologii, a nawet wizualizacji.

Budżet i oś czasu

Lokalizacja może być kosztowna i czasochłonna, zwłaszcza gdy mamy do czynienia z wieloma wyjściami i językami. Na początku należy oszacować:

- Koszt tłumaczenia w przeliczeniu na słowo
- Czas na cykl tłumaczenia (tworzenie → tłumaczenie → weryfikacja → publikacja)

- Zasoby potrzebne do wewnętrznej kontroli jakości i weryfikacji krajowej

Rozważ zaplanowanie buforów na zmiany treści pod koniec cyklu lub informacje zwrotne od recenzentów lokalizacji.

Wybór odpowiedniego przepływu pracy i narzędzi

Wybór właściwego przepływu pracy jest jedną z najważniejszych decyzji na etapie planowania. Czy będziesz używać:

- MadCap Lingo do zintegrowanego tłumaczenia i zarządzania projektami?
- Pliki wyjściowe MadCap Flare?
- Narzędzie CAT innej firmy z eksportem XLIFF z Lingo?

Decyzje te będą miały wpływ na strukturę folderów, rozwój tematów i warunkowe zarządzanie treścią w dalszej części procesu.

Ustalenie kryteriów tłumaczenia

Ustal wymierne kryteria dotyczące tego, jakie treści powinny, a jakie nie powinny być tłumaczone:

- Informacje prawne lub specyficzne dla regionu
- Osadzone media
- Elementy interfejsu użytkownika i zrzuty ekranu
- Treści ponownie wykorzystywane w różnych produktach lub wersjach

Zapewnia to opłacalną lokalizację i pozwala uniknąć marnowania zasobów na niepotrzebne lub mało istotne treści.

Dokumentacja

Prowadzenie przejrzystej dokumentacji dla:

- Struktura projektu i konwencje nazewnictwa
- Wymagania dotyczące tłumaczenia
- Punkty kontaktowe dla tłumaczy i recenzentów
- Procedury weryfikacji i zatwierdzania

Przejrzysta dokumentacja poprawia spójność i przyspiesza wdrażanie nowych współpracowników lub tłumaczy.

Struktura projektu

Odpowiednia struktura projektu od samego początku oszczędza czas, zmniejsza liczbę błędów i umożliwia ponowne wykorzystanie treści w różnych językach, wersjach wyjściowych i na różnych platformach.

Projekty nadrzędne dla współdzielonej zawartości

Jeśli wiele projektów ma wspólną zawartość, warto rozważyć utworzenie projektu nadrzędnego, który zawiera te wspólne elementy. Inne projekty mogą następnie odwoływać się do tego projektu nadrzędnego, zapewniając spójność i upraszczając aktualizacje zestawów dokumentacji.

Logiczna hierarchia folderów

Zorganizuj swoją zawartość w sposób, który odzwierciedla jej logiczny przepływ i relacje. Grupuj powiązane tematy w folderach i używaj opisowych konwencji nazewnictwa. Taka organizacja pomaga w nawigacji, zarówno autorom, jak i użytkownikom końcowym, oraz upraszcza zarządzanie treścią.

Foldery lub projekty językowe

W przypadku treści wielojęzycznych należy rozważyć sposób strukturyzacji wersji językowych:

- Pojedynczy projekt z tagami warunków i zmiennymi językowymi: Bardziej skomplikowany w zarządzaniu, ale wydajny w przypadku udostępnianych treści.
- Oddzielne projekty dla każdego języka: Łatwiejsze do wyizolowania tłumaczeń, ale trudniejsze do utrzymania spójności.

Flare pozwala na zarządzanie wieloma obiektami docelowymi z jednego projektu, co może uprościć konserwację, jeśli zostanie odpowiednio skonfigurowane.

Standaryzacja konwencji nazewnictwa

Spójne nazewnictwo tematów, fragmentów, warunków i zmiennych pomaga usprawnić tłumaczenie i uniknąć nieporozumień. Na przykład:

- Używaj prefiksów takich jak `t_`, `s_`, `img_` dla tematów, fragmentów i obrazów.
- Uwzględnij kody językowe w przetłumaczonych wersjach (na przykład `home_fr.htm` dla języka francuskiego).

Ta standaryzacja pomaga również w przypadku korzystania ze skryptów lub narzędzi tłumaczeniowych w celu zautomatyzowania części przepływu pracy.

Używaj spisów treści i celów w przemyślany sposób

Zdefiniuj różne spisy treści (TOC) i cele dla każdej grupy odbiorców lub języka. Pozwala to na:

- Dostosowane struktury nawigacji dla każdego języka lub produktu
- Warunkowe włączanie tematów
- Szczegółową kontrolę nad wyjściami (na przykład wyłączenie wyjścia PDF dla niektórych języków).

Przygotowanie do funkcji lokalizacji

Użyj wbudowanych funkcji Flare, aby umożliwić lokalizację od samego początku:

- "Zmienne" na stronie107: Dla brandingu, nazw produktów, tekstu prawnego - przetłumaczone raz i ponownie wykorzystane w całym projekcie.
- "Snippety" na stronie104: Idealne dla treści, które się powtarzają - definicja, ostrzeżenie lub zastrzeżenie.
- "Warunki" na stronie110: Do zarządzania widocznością treści według języka lub wyjścia.

Prowadzenie dokumentacji

Zachowaj dokładną dokumentację struktury projektu, konwencji i przepływów pracy. Ta praktyka jest nieoceniona przy wdrażaniu nowych członków zespołu, zapewnianiu ciągłości i wspieraniu wysiłków związanych z lokalizacją.

Single sourcing

Zarówno w przypadku tworzenia treści, jak i lokalizacji, kluczowe znaczenie ma unikanie redundancji i maksymalizacja wydajności. Jedną z najpotężniejszych strategii osiągnięcia tego celu w MadCap Flare jest single sourcing.

Single sourcing to praktyka polegająca na pisaniu i zarządzaniu treścią raz, a następnie ponownym jej wykorzystaniu lub zmianie przeznaczenia w wielu kontekstach, takich jak różne wyjścia, odbiorcy lub wersje produktu, bez powielania materiału źródłowego. Takie podejście nie tylko zwiększa spójność i zmniejsza wysiłek związany z utrzymaniem, ale także zapewnia znaczną przewagę podczas lokalizacji.

> Co ciekawe, single sourcing wykazuje podobieństwa koncepcyjne z pamięcią tłumaczeniową (Translation Memory, TM) wykorzystywaną w procesie lokalizacji. TM to baza danych, która przechowuje wcześniej przetłumaczone segmenty (zdania, akapity lub frazy). Gdy ta sama lub podobna treść pojawia się ponownie, system tłumaczeniowy sugeruje zapisane tłumaczenie zamiast tłumaczyć je od zera. W podobny sposób pojedyncze pozyskiwanie pozwala autorom na ponowne wykorzystanie istniejących bloków treści, zmniejszając liczbę segmentów, które należy utworzyć, zweryfikować lub przetłumaczyć.

Łącząc strategie pojedynczego pozyskiwania we Flare z korzyściami pamięci tłumaczeniowej w narzędziach takich jak MadCap Lingo lub narzędziach CAT innych firm, organizacje mogą znacznie obniżyć koszty tłumaczeń, czas realizacji i potencjalną niespójność między językami i wersjami.

Dlaczego single sourcing ma znaczenie dla lokalizacji

Podczas przygotowywania treści do tłumaczenia, każdy unikalny segment zwiększa koszty i czas. Powtarzające się, podobne treści muszą być nadal przetwarzane i sprawdzane, chyba że istnieje odpowiednia strategia ponownego wykorzystania. Bez single sourcingu nawet niewielkie aktualizacje współdzielonych treści wymagają ręcznego śledzenia i ponownego tłumaczenia. Single sourcing zapewnia, że:

- Piszesz (i tłumaczysz) treść tylko raz.
- Możesz dokonywać globalnych aktualizacji bloków treści, wpływając na każdą instancję, w której są one używane.
- Zlokalizowane wersje pozostają spójne w różnych produktach.

Kluczowe funkcje pojedynczego pozyskiwania w Flare

Tworzenie oparte na tematach

Model treści Flare opiera się na indywidualnych tematach, a nie na monolitycznych dokumentach. Każdy temat reprezentuje samodzielny fragment informacji, taki jak procedura, koncepcja lub odniesienie, które można połączyć w wiele materiałów wyjściowych (takich jak pliki PDF lub pomoc online). Dla lokalizacji:

- Zapewnia to, że tylko zaktualizowane tematy muszą być wysyłane do tłumaczenia.
- Tłumacze mogą pracować z mniejszymi, łatwiejszymi w zarządzaniu jednostkami treści.
- Tematy mogą być ponownie wykorzystywane w różnych produktach, platformach i zestawach dokumentacji.

Snippety

Snippety to fragmenty treści wielokrotnego użytku, takie jak powtarzające się ostrzeżenia dotyczące bezpieczeństwa, zastrzeżenia lub standardowe instrukcje. Zamiast kopiować i wklejać tę samą treść w różnych tematach, można wstawić fragment. Ma to wiele zalet:

- Gdy zmienia się fragment źródłowy, każda jego instancja jest automatycznie aktualizowana.
- Snippety zmniejszają ilość treści do przetłumaczenia.
- Tłumacze muszą przetłumaczyć snippet tylko raz, niezależnie od tego, ile razy się pojawi.

Przykładowe zastosowania snippetów:

- Bloki tekstu prawnego
- Typowe kroki rozwiązywania problemów
- Nagłówki i stopki
- Opisy interfejsu użytkownika

Zmienne

Zmienne działają jako symbole zastępcze dla tekstu, który może się różnić w różnych wyjściach lub językach, takich jak nazwy produktów, nazwy firm, numery wersji lub flagi funkcji.

Na przykład:

- `ProductName` = "Flare"
- `VersionNumber` = "2024.1"

Używanie zmiennych:

- Zapewnia spójność całej treści.
- Upraszcza tłumaczenie - tłumacze nie muszą się martwić, że terminy dotyczące produktów lub marek nieoczekiwanie się zmienią lub popełnią literówkę.
- Upraszcza aktualizacje - zmieniasz tylko wartość zmiennej, a nie każde wystąpienie w treści.

> Korzyści lokalizacyjne: Tylko definicje zmiennych muszą być przetłumaczone raz na język, a nie każde wystąpienie w dokumencie.

Tekst warunkowy

Tekst warunkowy umożliwia dołączanie lub wykluczanie treści na podstawie zdefiniowanych warunków (takich jak wersja produktu, region lub grupa odbiorców). Jest to szczególnie przydatne w wielojęzycznych lub wieloproduktowych scenariuszach dokumentacji i eliminuje potrzebę oddzielnych projektów dla każdej odmiany, zmniejszając duplikację i poprawiając skalowalność.

Zarządzanie celami

Cele we Flare definiują ustawienia wyjściowe, takie jak format (HTML5, PDF, Word), język, zestawy warunków, definicje zmiennych i arkusze stylów. Każdy cel może generować dostosowaną wersję dokumentacji, a wszystko to z jednego źródła.

Podsumowanie korzyści

Korzyści	Opis
Spójność	Wspólna treść, terminologia i odniesienia do interfejsu użytkownika pozostają jednolite we wszystkich dokumentach i językach.
Niższe koszty tłumaczenia	Tłumacze pracują tylko nad nowymi lub zaktualizowanymi treściami; treści wielokrotnego użytku nie muszą być ponownie tłumaczone.
Krótszy czas wprowadzenia produktu na rynek	Lokalizacja może przebiegać szybciej, gdy jest mniej unikalnych treści do przetworzenia.

Korzyści	Opis
Uproszczona konserwacja	Aktualizacje wprowadzone do pojedynczego tematu, fragmentu lub zmiennej są kaskadowane na wszystkich wyjściach.
Skalowalność	Obsługa rosnących bibliotek treści i wielojęzycznych rozszerzeń bez wykładniczego wzrostu nakładu pracy.

Najlepsze praktyki dla efektywnego pojedynczego pozyskiwania

- Zaplanuj ponowne wykorzystanie na początku: Zidentyfikuj współdzielone bloki treści i zaprojektuj tematy tak, aby były modułowe i niezależne.
- Unikaj osadzania treści: Nie pisz długich, ciągłych narracji, które trudno oddzielić. Podziel treści na łatwe do zarządzania, ukierunkowane tematy.
- Dokumentuj swoją strategię dotyczącą treści: Przechowuj jasne zapisy dotyczące sposobu używania zmiennych, fragmentów i warunków, aby tłumacze i przyszli autorzy mogli zrozumieć system.
- Często testuj wyniki: Upewnij się, że wszystkie pliki docelowe (zlokalizowane i nie tylko) kompilują się poprawnie z odpowiednimi warunkami treści i definicjami zmiennych.
- Przeszkol swoich dostawców usług lokalizacyjnych: Upewnij się, że Twój zespół tłumaczy rozumie użycie fragmentów i zmiennych, aby nie zepsuć komponentów wielokrotnego użytku podczas tłumaczenia.

Pisanie pod kątem lokalizacji

Tworzenie treści możliwych do zlokalizowania jest kluczowym elementem tworzenia przyjaznej dla użytkownika dokumentacji, która obsługuje odbiorców z całego świata. Podczas gdy tłumaczenie zajmuje się konwersją treści z jednego języka na drugi, lokalizacja jest bardziej holistyczna – dostosowuje treść tak, aby odzwierciedlała niuanse kulturowe, normy regulacyjne i oczekiwania dotyczące formatowania specyficzne dla każdej lokalizacji. W tej sekcji opisano strategie i praktyki dotyczące pisania treści gotowych do lokalizacji od samego początku, co znacznie poprawia jakość i zmniejsza koszty tłumaczenia oraz czas realizacji.

Ogólne wytyczne

Podstawą przyjaznej lokalizacji treści jest przejrzystość, spójność i prostota. Treść źródłowa powinna być łatwo zrozumiała dla tłumaczy i ustrukturyzowana, aby uniknąć niejednoznaczności, błędnej interpretacji lub niepotrzebnej złożoności.

Kluczowe kwestie:

- Twórz treści, zakładając, że zostaną przetłumaczone na wiele języków.
- Utrzymuj spójność terminologii w całej dokumentacji.
- Używaj narzędzi, takich jak przewodniki stylistyczne, glosariusze i szablony, aby ujednolicić strukturę i ton.
- Zaplanuj tłumaczenie od samego początku, nawet jeśli lokalizacja nie jest natychmiast wymagana.

Koncentrując się na wysokiej jakości treści źródłowej, organizacje przygotowują grunt pod dokładną i wydajną lokalizację na późniejszym etapie cyklu życia projektu.

Język i struktura zdań

Jasny, prosty język jest kluczowy dla tworzenia treści przyjaznych lokalizacji. Sposób konstruowania zdań – i używany język – bezpośrednio wpływa nie tylko na to, jak dobrze treść jest rozumiana przez odbiorców na całym świecie, ale także na to, jak łatwo i dokładnie można ją przetłumaczyć, zwłaszcza przy użyciu systemów tłumaczenia maszynowego lub pamięci tłumaczeniowej.

Dlaczego prosty język ma znaczenie w lokalizacji

Gdy treść źródłowa jest napisana jasno i prosto:

- Tłumacze mogą ją zrozumieć szybciej i dokładniej.
- Narzędzia do tłumaczenia maszynowego (MT) dają lepsze wyniki.
- Koszty lokalizacji są niższe ze względu na mniejszą liczbę niejasności lub potrzeb wyjaśnień.
- Użytkownicy końcowi mogą szybko zrozumieć informacje, niezależnie od pochodzenia kulturowego lub językowego.

> Pisanie prostym językiem nie polega na „spłycaniu" treści – chodzi o to, aby informacje były jasne, dostępne i możliwe do przetłumaczenia. Zmniejsza to nieporozumienia, skraca czas lokalizacji i poprawia doświadczenia użytkowników na całym świecie.

Pisanie prostym, globalnym językiem angielskim

Używaj jasnego, prostego słownictwa. Preferuj codzienne terminy zamiast technicznego żargonu lub idiomatycznych wyrażeń. Wybieraj jednoznaczny język, aby zmniejszyć ryzyko błędnego tłumaczenia lub nieporozumienia.

- Unikaj idiomów i metafor

  ✗*Dopracowujemy szczegóły.*
  ✓ *Finalizujemy szczegóły.*

- Unikaj slangu i zwrotów specyficznych dla regionu

  ✗*Spróbuj.*
  ✓ *Spróbuj.*

- Stosuj standardową pisownię i gramatykę języka angielskiego

  Trzymaj się spójnych konwencji pisowni (na przykład amerykańskiego lub brytyjskiego) i unikaj przełączania się w ramach projektu.

Używaj krótkich, bezpośrednich zdań

Długość i struktura zdania mają bezpośredni wpływ na możliwość tłumaczenia. Krótsze zdania są:

- Łatwiejsze do zrozumienia
- Łatwiejsze do analizy przez silniki tłumaczeń maszynowych
- Łatwiejsze do segmentowania w systemach pamięci tłumaczeniowej

Najlepsze praktyki:

- Jeśli to możliwe, utrzymuj zdania poniżej 20 słów.
- Rozbijaj długie, złożone zdania na oddzielne pomysły.
- Wyeliminuj niepotrzebne modyfikatory lub kwalifikatory.

✗*Jeśli opcja jest włączona i wszystkie wymagania systemowe są spełnione, użytkownik będzie mógł kontynuować korzystanie z oprogramowania bez dalszych przerw.*
✓ *Włącz opcję. Jeśli wymagania systemowe są spełnione, oprogramowanie działa bez przerw.*

Używaj strony czynnej

Strona czynna jest wyraźniejsza i bardziej bezpośrednia, co ułatwia zrozumienie i ułatwia tłumaczom zapamiętanie znaczenia.

✗*System powinien zostać sprawdzony przez użytkownika.*
✓ *Użytkownik powinien sprawdzić system.*

Pomaga to również wyjaśnić, kto co robi, zmniejszając niejednoznaczność – zwłaszcza w treściach proceduralnych lub instruktażowych.

Używaj spójnej terminologii

Używaj tego samego terminu, aby odnosić się do tej samej koncepcji w całej dokumentacji. Niespójne nazewnictwo zwiększa obciążenie poznawcze czytelników i tworzy dodatkową pracę dla tłumaczy, którzy mogą musieć zweryfikować, czy dwa różne terminy odnoszą się do tej samej rzeczy.

- Prowadź słownik specyficzny dla projektu.
- Zdefiniuj preferowane terminy w swoim przewodniku stylistycznym.

- Używaj zmiennych MadCap Flare dla nazw produktów, numerów wersji lub innych powtarzających się terminów, które mogą się zmieniać między lokalizacjami.

Unikaj dwuznaczności

Niejednoznaczny język powoduje zamieszanie w każdym języku. Podczas tłumaczenia treści dwuznaczność jest wzmacniana.

✕*Kliknij przycisk, gdy się pojawi.*
✓ *Kliknij przycisk Instaluj, gdy pojawi się na pasku narzędzi.*

Bądź konkretny w odniesieniu do:

- Obiektów („to okno dialogowe” kontra „okno dialogowe potwierdzenia”)
- Akcji
- Warunków i czasu

Używaj równoległych struktur zdań

Używaj spójnej składni i struktury dla podobnej treści, aby ułatwić czytelność i ponowne wykorzystanie pamięci tłumaczeniowej.

✓ Dobrze:

- *Kliknij ikonę Zapisz.*
- *Kliknij ikonę Drukuj.*
- *Kliknij ikonę Eksportuj.*

✕Niespójne:

- *Kliknij ikonę Zapisz.*
- *Aby wydrukować dokument, wybierz ikonę Drukuj.*
- *Wyeksportuj plik za pomocą opcji Eksportuj.*

Preferuj standardową kolejność wyrazów

Używaj struktury Podmiot-Orzeczenie-Dopełnienie (SVO). Jest to najczęstszy format zdania i jest dobrze obsługiwany przez narzędzia MT.

✓ *Użytkownik wprowadza hasło.*

✗*Użytkownik musi wprowadzić hasło.*

Rozważ użycie tłumaczenia maszynowego (MT) i pamięci tłumaczeniowej (TM)

Proste, przewidywalne wzorce zdań są korzystne dla systemów tłumaczeniowych. Unikaj kreatywnych lub złożonych fraz, które mogą zepsuć dopasowania w pamięci tłumaczeniowej lub obniżyć jakość MT.

> Pisz tak, jakby maszyna przeczytała to jako pierwsza – ponieważ tak zrobi.

Unikaj również:

- Długich łańcuchów rzeczowników: „preferencje ostrzeżenia o wygaśnięciu hasła użytkownika”
- Dwuznaczności zaimków: „Jeśli się nie powiedzie, uruchom ponownie”. (Co się nie powiedzie?)

Pamiętaj o odbiorcach

Pisanie pod kątem lokalizacji oznacza pisanie dla osób, dla których dany język nie jest językiem ojczystym, i użytkowników z różnych kultur. Używaj języka inkluzywnego i pełnego szacunku oraz unikaj założeń dotyczących poziomu technicznego, pochodzenia lub środowiska użytkownika.

Łagodzenie błędów

Błędy w projektach lokalizacyjnych często mają swoje źródło w treści źródłowej. Mogą się one rozprzestrzeniać podczas tłumaczenia i wymagają kosztownej edycji i zapewnienia jakości.

Najlepsze praktyki w celu zmniejszenia liczby błędów w tekście:

- Sprawdź poprawność językową: Literówki, brakujące artykuły lub niejasne sformułowania mogą dezorientować zarówno czytelników, jak i tłumaczy.
- Zminimalizuj tekst w materiałach multimedialnych: Unikaj osadzania tekstu w obrazach lub animacjach, co wymaga ponownego tworzenia w każdym języku.
- Obsługuj symbole zastępcze i zmienne w sposób spójny: Na przykład zawsze używaj spójnej notacji dla nazw produktów lub terminów interfejsu użytkownika.
- Wstępna walidacja treści: Użyj wbudowanych narzędzi analitycznych MadCap Flare, aby wykryć problemy z formatowaniem, uszkodzone łącza lub zduplikowane identyfikatory przed eksportem.

Przewodniki stylistyczne i glosariusze

Ustandaryzowanie sposobu, w jaki zespół pisze i tłumaczy terminologię, jest niezbędne do zachowania spójności, czytelności i profesjonalnej jakości wszystkich zlokalizowanych treści. W projektach wielojęzycznych niespójność może podważyć zaufanie użytkowników, wprowadzić czytelników w błąd i zwiększyć koszty tłumaczeń z powodu możliwych do uniknięcia przeróbek. W tym miejscu przewodniki stylistyczne i glosariusze stają się niezbędne.

Dlaczego przewodnik stylistyczny jest ważny

Dobrze ustrukturyzowany przewodnik stylistyczny służy jako dokument referencyjny dla autorów, redaktorów i tłumaczy. Definiuje sposób pisania i formatowania treści zarówno w języku źródłowym, jak i docelowym. Do najważniejszych korzyści należą:

- Wymuszanie spójności: Zapewnia ten sam ton, strukturę zdań i terminologię w różnych tematach, przez różnych autorów i w różnych językach.
- Porady dotyczące gramatyki i tonu: Wyjaśnia, czy treść powinna być formalna czy konwersacyjna, aktywna czy bierna, i podaje przykłady preferowanych konstrukcji.
- Standardy formatowania: Określa sposób formatowania:
    - Jednostki miary (na przykład metryczne kontra imperialne)
    - Numeracja i style punktowania
    - Zasady kapitalizacji
    - Znaki cudzysłowu (na przykład "angielski" kontra „polski”)
    - Separatory dziesiętne (na przykład 1,000.00 kontra 1.000,00)
- Wyrównanie odbiorców i celu: Definiuje, dla kogo jest przeznaczona treść (na przykład dla techników czy użytkowników końcowych), pomagając autorom odpowiednio dostosować złożoność i ton.
- Uwagi dotyczące konkretnego języka: Przewodniki stylistyczne dostosowane do konkretnego języka mogą zawierać instrukcje dotyczące języka z podziałem na płeć, formalnego zwracania się i preferencji dotyczących długości zdań.

Glosariusze do zarządzania terminologią

Glosariusz to coś więcej niż lista słów – to kontrolowany słownik, który pokazuje zatwierdzone tłumaczenia kluczowych terminów. Jest to szczególnie ważne w przypadku:

- Terminów technicznych: Zapewnia spójność w złożonych domenach, takich jak oprogramowanie, inżynieria czy opieka zdrowotna.
- Ciągi i etykiety interfejsu użytkownika: Dopasowuje sformułowania używane w interfejsie oprogramowania, dzięki czemu użytkownicy nie są zdezorientowani niezgodnościami między interfejsem użytkownika a dokumentacją.
- Terminy nieprzetłumaczalne: Identyfikuje nazwy marek, nazwy produktów, akronimy lub elementy kodu, które powinny pozostać w języku angielskim lub ich języku źródłowym.
- Unikanie fałszywych przyjaciół: Zawiera uwagi dotyczące terminów, których nie należy tłumaczyć dosłownie, aby uniknąć nieporozumień lub błędów kulturowych.

Korzyści dla przepływu pracy lokalizacji

- Zmniejsza liczbę zapytań i nieścisłości tłumacza.
- Poprawia wykorzystanie pamięci tłumaczeniowej (TM).
- Zapewnia lepsze wyniki QA przy mniejszej liczbie cykli rewizji.
- Przyspiesza wdrażanie nowych lingwistów lub członków zespołu.

Akceptuj zmiany

Przed wysłaniem treści do tłumaczenia, kluczowe jest zaakceptowanie wszystkich śledzonych zmian w projekcie MadCap Flare lub dokumentach źródłowych. Chociaż śledzone zmiany są pomocne podczas przeglądu, pozostawienie ich na miejscu może powodować poważne problemy z lokalizacją.

Dlaczego to ważne

- Zawyżenie liczby słów: Zarówno usunięty, jak i wstawiony tekst mogą być liczone, co niepotrzebnie zwiększa koszty tłumaczenia.
- Dezorientacja tłumacza: Nie jest jasne, która wersja tekstu jest ostateczna, co prowadzi do opóźnień lub błędów.
- Zgodność narzędzi: Narzędzia CAT (w tym MadCap Lingo) mogą nie obsługiwać prawidłowo śledzonych zmian, co może powodować problemy z formatowaniem lub segmentacją.

Najlepsze praktyki

- Finalizuj edycje przed eksportem: Upewnij się, że przeglądy są kompletne, a treść jest stabilna.
- Akceptuj lub odrzucaj wszystkie zmiany: Użyj panelu Przegląd Flare, aby potwierdzić czysty stan.
- Eksportuj tylko wersję sfinalizowaną: Dzięki temu tłumacze otrzymują spójne, jednoznaczne dane wejściowe.

Komunikacja z LSP

Skuteczna komunikacja z dostawcą usług językowych (LSP) ma kluczowe znaczenie dla pomyślnej lokalizacji. Ustanowienie jasnych, spójnych i proaktywnych kanałów komunikacji zapewnia dokładne przetłumaczenie i lokalizację treści, zgodnie z celami i harmonogramami projektu.

Nawiązanie współpracy

- Wczesne zaangażowanie: Zaangażuj swojego LSP na etapie planowania projektu. Wczesna współpraca umożliwia lepsze dopasowanie zakresu projektu, harmonogramów i oczekiwań.
- Zrozumienie możliwości: Upewnij się, że Twój LSP zna MadCap Flare i powiązane z nim narzędzia, takie jak MadCap Lingo i MadCap Capture, lub dowiedz się, które narzędzie innej firmy zostanie użyte do lokalizacji Twojego projektu.

Podaj kompleksowy kontekst

- Przegląd projektu: Udostępnij szczegółowe informacje o strukturze, celach i grupie docelowej swojego projektu. Obejmuje to zapewnienie dostępu do przewodników po stylach, glosariuszy i wszelkich istniejących pamięci tłumaczeniowych.
- Materiały referencyjne: Dostarcz swojemu LSP odpowiednie materiały, takie jak ostateczne pliki PDF, zrzuty ekranu i przewodniki szkoleniowe. Te zasoby oferują kontekst, który pomaga w dokładnym tłumaczeniu.

Zdefiniuj jasne procesy i oczekiwania

- Dopasowanie przepływu pracy: Omów i uzgodnij przepływy pracy, w tym formaty plików, harmonogramy dostaw i procedury zapewniania jakości. Przejrzyste przepływy pracy minimalizują nieporozumienia i usprawniają proces lokalizacji.
- Mechanizmy informacji zwrotnej: Ustanów kanały regularnych informacji zwrotnych i aktualizacji. Otwarta komunikacja umożliwia szybkie rozwiązywanie problemów i ciągłe doskonalenie.

Wykorzystaj technologię i narzędzia

- Wykorzystywanie narzędzi CAT: Niezależnie od tego, czy używasz MadCap Lingo, czy innych narzędzi CAT, takich jak Trados Studio, memoQ lub Phrase (Memsource), upewnij się, że Twój LSP jest biegły w wybranej platformie. Zgodność z eksportami Flare (na przykład XLIFF, HTML5 lub PDF) jest niezbędna do utrzymania

dokładności i wydajności tłumaczenia.

- Pamięć tłumaczeniowa (TM) i zarządzanie terminologią: Współpracuj ze swoim LSP w celu tworzenia i utrzymywania solidnych TM i baz terminologicznych. Te zasoby promują spójność tłumaczeń i obniżają koszty w czasie.

Ciągła współpraca i doskonalenie

- Regularne przeglądy: Przeprowadzaj okresowe przeglądy przetłumaczonych treści, aby zapewnić jakość i zgodność z wytycznymi. Wspólne przeglądy wzmacniają poczucie partnerstwa i wspólnej odpowiedzialności.
- Szkolenia i aktualizacje: Informuj swojego LSP o aktualizacjach swoich produktów lub usług. Regularne sesje szkoleniowe mogą pomóc Twojemu LSP pozostać w zgodzie z Twoimi zmieniającymi się potrzebami.

Rozważne korzystanie z funkcji Flare

MadCap Flare oferuje rozbudowany zestaw funkcji do tworzenia treści z jednego źródła, treści modułowych i gotowych do lokalizacji. Jednak nieostrożne lub niespójne korzystanie z tych funkcji może utrudnić zarządzanie projektami i znacznie zwiększyć nakład pracy i koszty lokalizacji. W tej sekcji opisano, jak skutecznie korzystać z niektórych z najpotężniejszych funkcji Flare – fragmentów kodu, zmiennych, warunków i linków – w sposób, który wspiera płynną i wydajną lokalizację.

Snippety

Czym są:

Snippety w MadCap Flare to wielokrotnego użytku bloki treści, które można wstawiać do wielu tematów. Są jak klocki – jeśli raz utworzysz fragment treści (np. ostrzeżenie, notatkę lub powtarzający się krok procedury), możesz go ponownie wykorzystać w całej dokumentacji, wstawiając go jako fragment.

Fragment zachowuje się jak żywe odniesienie: gdy edytujesz plik fragmentu, wszystkie jego wystąpienia są automatycznie aktualizowane, gdziekolwiek został wstawiony.

Dlaczego fragmenty są ważne dla lokalizacji

Podczas przygotowywania treści do tłumaczenia powtórzenie równa się kosztowi. Powtarzanie tej samej treści w różnych plikach może zwiększyć liczbę słów, czas tłumaczenia i ryzyko niespójnych tłumaczeń. Fragmenty pomagają wyeliminować to poprzez:

- Zmniejszanie nadmiarowej treści, która musi zostać przetłumaczona.
- Zapewnianie spójnego frazeologizmu i struktury we wszystkich zlokalizowanych wynikach.
- Ułatwianie aktualizacji, zwłaszcza gdy zmiany występują pod koniec cyklu dokumentacji.

Jeśli fragment został już raz przetłumaczony, większość narzędzi CAT rozpozna go i ponownie wykorzysta istniejące tłumaczenie, oszczędzając czas i pieniądze.

Najlepsze praktyki dotyczące lokalizacji:

- Używaj w przypadku modułowej, powtarzalnej treści: Fragmenty zmniejszają liczbę słów wymagających tłumaczenia i utrzymują spójność treści. Na przykład standardowe wyłączenie odpowiedzialności używane w 30 tematach musi zostać przetłumaczone tylko raz, jeśli zostanie ponownie użyte za pośrednictwem fragmentu.
- Nie używaj w zdaniach częściowych: Tłumacze nie widzą całego zdania, każda część jest przechowywana osobno. Każdy fragment jest osobnym plikiem na długiej liście innych plików, co utrudnia tłumaczom poznanie jego kontekstu. Ponadto treść fragmentu może zależeć od części zdania, która nie jest podzielona na fragmenty, na przykład w językach fleksyjnych, takich jak polski.
- Utrzymuj fragmenty w stanie zamkniętym: Unikaj dołączania pełnych akapitów lub treści zależnych od kontekstu. Fragmenty nie powinny zależeć od treści zewnętrznej, aby można je było zrozumieć w tłumaczeniu.

- Unikaj nadmiernego zagnieżdżania: Głęboko zagnieżdżone fragmenty (fragmenty wewnątrz fragmentów) komplikują zarówno proces tworzenia, jak i tłumaczenia. Ogranicz do jednego poziomu, jeśli to możliwe.

✓ Dobry przykład:

```
<Snippet>Note: This feature is only available in the Pro version.</Snippet>
```

Dlaczego jest dobry:

- Ponownego użytku, przejrzysty i niezależny od kontekstu.
- Łatwy do zlokalizowania raz i ponownego użycia globalnie.
- Brak osadzonego formatowania lub warunków.

✗Kiepski przykład:

```
<Snippet>The error occurs if </Snippet><Variable>[data element]</Variable><Snippet>
is populated, but source is empty or null.</Snippet>
```

Problemy:

- Tłumacz widzi oba fragmenty jako osobne pliki. Prawdopodobnie wyglądają w sposób, w jaki tłumacz nie może rozpoznać, że należą do siebie.
- W niektórych językach struktura gramatyczna kolejności fragmentów może wymagać zmiany.

> Porada: W Flare i Lingo możesz spłaszczyć fragmenty i przekonwertować je na zwykły tekst. W ten sposób tłumacz widzi pełne konteksty i może za każdym razem dostosować strukturę gramatyczną.

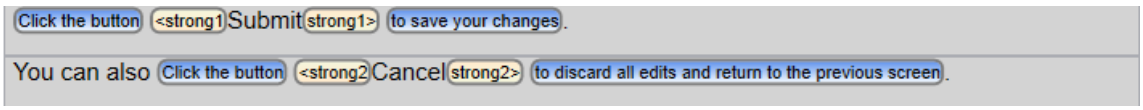
Punkt widzenia tłumacza

Podzielenie zdania na wiele fragmentów przerywa płynność zdania i usuwa niezbędny kontekst dla tłumaczy. Tak może wyglądać tekst z nieprawidłowo zastosowanymi fragmentami w MadCap Flare:

[ClikTheButtonClick the button] [Submit] [ToSaveChangto save your changes] You can also [ClikTheButtonClick the button] [Cancel] [ToDiscardChto discard all edits and return to the previous screen].

Tak widzi ten tekst tłumacz z zachowanymi fragmentami:

- MadCap Lingo



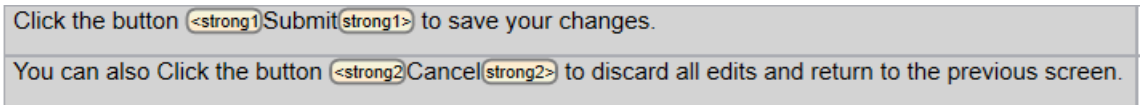
- SDL Trados Studio

Problemy:

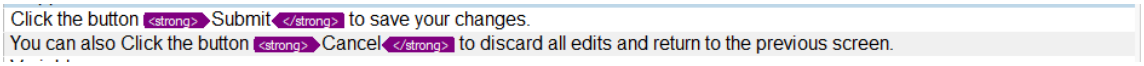
- Tekst jest fragmentaryczny i pozbawiony kontekstu.
- MadCap Lingo pokazuje zawartość fragmentu; jednak nie można jej od razu edytować, ponieważ fragment jest przechowywany w osobnym pliku.
- W narzędziach CAT innych firm, takich jak SDL Trados Studio lub memoQ, tłumacz nie widzi od razu zawartości fragmentu. Musi ją znaleźć ręcznie w plikach projektu.
- Dostosowanie struktury gramatycznej zdania jest niemal niemożliwe ze względu na stałą naturę fragmentów.

MadCap Flare daje możliwość spłaszczenia fragmentów podczas procesu eksportu w celu rozwiązania niektórych możliwych problemów. Oto jak tłumacz widzi tekst ze spłaszczonymi fragmentami:

- MadCap Lingo



- SDL Trados Studio



W przypadku fragmentów spłaszczenie ich rozwiązuje większość problemów. Jednak funkcja fragmentów jest tracona w przetłumaczonych plikach, podobnie jak korzyści z ich używania.

Zmienne

Czym są:

Zmienne są symbolami zastępczymi dla treści, które mogą się zmieniać w czasie lub między wynikami, takich jak nazwy produktów, numery wersji lub nazwy firm.

Najlepsze praktyki dotyczące lokalizacji:

- Używaj tylko dla nazw własnych, wersji i innych nieprzetłumaczalnych tekstów: Nazwy produktów, numery wersji i branding powinny pozostać spójne i nieprzetłumaczone. Nigdy nie używaj ich do ogólnych terminów, takich jak „instrukcja”, „strona internetowa”, „urządzenie”.
- Unikaj używania zmiennych w środku zdania dla słów, które można przetłumaczyć: Tłumacze potrzebują pełnego kontekstu, a segmentowane frazy ze zmiennymi utrudniają prawidłową strukturę zdań i gramatykę w innych językach.
- Używaj opisowych konwencji nazewnictwa: Używaj nazw takich jak `ProductName_Pro`, a nie tylko `Var1`, aby pomóc tłumaczom i innym autorom zrozumieć treść.

✓ Dobry przykład:
The `<Variable>ProductName</Variable>` supports multiple file formats.

Dlaczego jest dobry:

- Jasna, opisowa nazwa
- Zawiera rzeczownik własny, który nie zależy od struktury gramatycznej zdania.

✕Zły przykład:
This `<Variable>EventType</Variable>` has already ended. You can start a new one.

- gdzie `EventType` może być „webinar” lub „meeting”

Problemy:

W niektórych językach zmienna może zawierać słowa o różnych rodzajach (w języku polskim „webinar” jest rodzaju męskiego (ten webinar), a „meeting” jest rodzaju nijakiego (to spotkanie)). Ma to wpływ na całe zdanie:

- *This `webinar` has already ended. You can start a new one.* staje się *Ten webinar już się zakończył. Możesz rozpocząć nowy.*

- *This `meeting` has already ended. You can start a new one.* staje się *To spotkanie już się zakończyło. Możesz udzielić licencji nowe.*

> Wskazówka: W Flare i Lingo możesz spłaszczyć zmienne i przekonwertować je na zwykły tekst. W ten sposób tłumacz może za każdym razem dostosować strukturę gramatyczną.

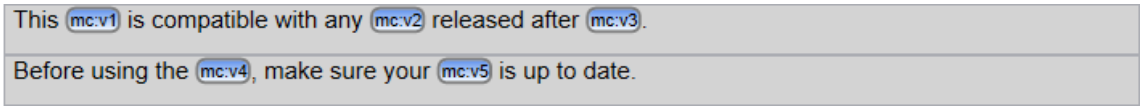
Punkt widzenia tłumacza

Używanie zmiennych dla rzeczowników pospolitych utrudnia – a nawet uniemożliwia – uwzględnienie różnych rodzajów, liczb i przypadków gramatycznych w tłumaczonej treści. Tak może wyglądać tekst z nieprawidłowo zastosowanymi zmiennymi w MadCap Flare:

This DeviceType: printer is compatible with any PlatformNam operating system released after ReleaseDate June 2024.
Before using the DeviceType: printer, make sure your PlatformNam operating system is up to date.

Oto jak tłumacz widzi ten tekst z zachowanymi zmiennymi:

- MadCap Lingo



- SDL Trados Studio

Problemy:

- Tłumacz nie widzi zawartości zmiennej – ani w MadCap Lingo, ani w narzędziu CAT innej firmy – ponieważ zmienne są przechowywane w oddzielnym pliku.
- Rzeczowniki pospolite używane jako zmienne zazwyczaj muszą zostać przetłumaczone, a w niektórych językach odmienione lub w inny sposób dostosowane gramatycznie. Nie jest to możliwe podczas tłumaczenia ze względu na stałą naturę zmiennych.

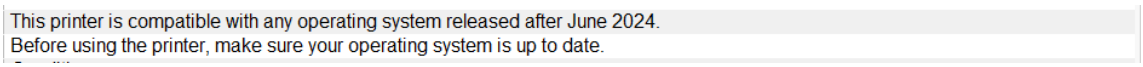
- Zmiana zmiennej może zmienić rodzaj rzeczownika, który zawiera w tłumaczonym języku. Taka zmiana może wymagać przepisania otaczającego zdania, co niweczy cel używania zmiennych w celu zaoszczędzenia czasu i wysiłku.

MadCap Flare daje możliwość spłaszczenia zmiennych podczas procesu eksportu w celu rozwiązania niektórych możliwych problemów. Oto jak tłumacz widzi tekst ze spłaszczonymi zmiennymi:

- MadCap Lingo

This printer is compatible with any operating system released after June 2024.
Before using the printer, make sure your operating system is up to date.

- SDL Trados Studio



W przypadku zmiennych, spłaszczenie ich rozwiązuje większość problemów. Jednak funkcja zmiennych jest tracona w przetłumaczonych plikach, podobnie jak korzyści z ich używania.

Warunki

Czym są:

Warunki w MadCap Flare to mechanizm filtrowania treści, który umożliwia tworzenie wielu wariantów dokumentacji z jednego źródła. Możesz stosować tagi warunków do tematów, akapitów lub obrazów, aby uwzględnić lub wykluczyć treść w zależności od celu wyjściowego – na przykład według języka, odbiorców, wersji produktu lub regionu.

Jest to niezbędne w przypadku pojedynczego źródła i obsługuje takie scenariusze, jak:

- Wyświetlanie różnych instrukcji dla różnych wydań oprogramowania (na przykład Standard vs. Pro)
- Ukrywanie treści specyficznych dla regionu w wynikach globalnych
- Wyświetlanie tylko treści istotnych dla odbiorców wewnętrznych vs. zewnętrznych

Jednak chociaż warunki są pomocne, należy nimi ostrożnie zarządzać podczas przygotowywania treści do lokalizacji.

Najlepsze praktyki dotyczące lokalizacji:

1. Ogranicz liczbę zestawów warunków

Posiadanie zbyt wielu tagów warunków lub złożonej logiki warunkowej może znacznie skomplikować tłumaczenie. Tłumacze mogą otrzymywać treści, nie wiedząc, która kombinacja warunków będzie widoczna razem, co zwiększa ryzyko pomyłki lub niespójności.

2. Używaj jasnych, znaczących nazw

Unikaj ogólnych lub tajemniczych nazw, takich jak `Cond1`, `Tag2` lub `OldNew`. Zamiast tego używaj spójnych, opisowych nazw, takich jak:

- `Language_FR`
- `Product_Pro`
- `Region_EU`
- `Audience_Admin`

Poprawia to łatwość utrzymania projektu i pomaga tłumaczom i recenzentom lepiej zrozumieć kontekst.

3. Unikaj nakładania się warunków w tym samym bloku treści

Nakładanie się wielu tagów warunków w tym samym akapicie lub zdaniu – zwłaszcza gdy są słabo udokumentowane – może prowadzić do nieprzewidywalnych kombinacji wyników i błędnych tłumaczeń. Zamiast tego wypisz całe zdania akapitów i przypisz im wymagane warunki.

4. Dokładnie udokumentuj użycie warunków

Utwórz mapę warunków lub przewodnik, który wyjaśnia:

- Które tagi są używane
- Na jaką treść wpływają
- Do jakich wyników lub ustawień regionalnych należą

Jest to szczególnie pomocne dla dostawców tłumaczeń pracujących poza Twoim środowiskiem autorskim.

✓ Dobry przykład:

```
<p MadCap:conditions="Audience_Admin">Only administrators can access the system settings via the Configuration panel.</p><p MadCap:conditions="Audience_EndUser">You can change your personal preferences in the Settings menu.</p>
```

Dlaczego jest to dobre:

- Warunki są stosowane na poziomie akapitu, dzięki czemu treść jest oddzielona i przejrzysta.
- Tagi są opisowe i oczywiste (`Audience_Admin`, `Audience_EndUser`), więc nie ma niejasności.
- Tłumacze mogą łatwo stwierdzić, która treść jest przeznaczona dla której grupy użytkowników, a Ty możesz generować wyniki specyficzne dla odbiorców bez pomyłek.

✕Kiepski przykład:

```
<p MadCap:conditions="UserType_Adv, Product_Old">This advanced feature is only present in the old version.</p>
```

Problemy:

- Wiele tagów warunkowych jest ułożonych w stos bez przejrzystości. Nie jest jasne, co będzie zawierać ostateczny wynik. Jeśli nie ma dokumentacji, tłumacz może nie wiedzieć, który wariant zostanie użyty lub czy przetłumaczyć oba.

Punkt widzenia tłumacza

Używanie warunków w środku zdania w celu utworzenia wariantów treści może sprawić, że nie będzie jasne, jak będą wyglądać ostateczne wyniki, a tłumaczom będzie trudniej z nimi pracować. Tak może wyglądać

Tekst z nieprawidłowo zastosowanymi warunkami w MadCap Flare:

The DM DEXF7 ozone destructor s is are designed for destructing process off-gas flows up to 25, 50, 75 or 100 7 m^3/h, respectively.

Tak tłumacz widzi ten tekst z warunkami:

- MadCap Lingo

The <mc:ct1>DM</mc:ct1><mc:ct2>DEXF7</mc:ct2> ozone destructor<mc:ct3>s</mc:ct3> <mc:ct4>is</mc:ct4><mc:ct5>are</mc:ct5> designed for destructing process off-gas flows up to <mc:ct6>25, 50, 75 or 100</mc:ct6><mc:ct7>7</mc:ct7> m^3/h<mc:ct9>, respectively</mc:ct9>.

- SDL Trados Studio

The <MadCap:conditionalText MadCap:conditions="Default.DMmodel" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">
DM</MadCap:conditionalText><MadCap:conditionalText MadCap:conditions="Default.DEXF7model" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">
DEXF7</MadCap:conditionalText> ozone
destructor<MadCap:conditionalText MadCap:conditions="Default.DMmodel" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">s</MadCap:conditionalText>
<MadCap:conditionalText MadCap:conditions="Default.DEXF7model" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">
is</MadCap:conditionalText><MadCap:conditionalText MadCap:conditions="Default.DMmodel" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">
are</MadCap:conditionalText> designed for destructing process off-gas flows up to
<MadCap:conditionalText MadCap:conditions="Default.DMmodel" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">25, 50, 75 or
100</MadCap:conditionalText><MadCap:conditionalText MadCap:conditions="Default.DEXF7model" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">7
</MadCap:conditionalText>
m³/h<MadCap:conditionalText MadCap:conditions="Default.DMmodel" xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd">,
respectively</MadCap:conditionalText>.

Problemy:

- Nadmiar tagów powoduje bałagan wizualny, utrudniając zrozumienie struktury zdania i przewidzenie ostatecznych wyników.
- Tekst warunkowy może wymagać różnych struktur gramatycznych w językach fleksyjnych, co utrudnia lub wręcz uniemożliwia dokładne tłumaczenie.
- Tworzenie form liczby mnogiej rzeczowników poprzez dodawanie „-s” wewnątrz znacznika warunkowego sprawia, że tłumaczenie jest niemal niemożliwe w wielu językach.

Hiperłącza kontra odsyłacze

Czym są:

- Hiperłącza: Bezpośrednie łącza do innych tematów lub witryn.
- Odnośniki (xref): Dynamiczne łącza, które zawierają tekst z obiektu docelowego (taki jak nagłówek lub tytuł) i są automatycznie aktualizowane po zmianie obiektu docelowego.

Dlaczego ma to znaczenie dla lokalizacji

W projektach wielojęzycznych odsyłacze są zazwyczaj preferowane, ponieważ:

- Dostosowują się do przetłumaczonych nagłówków: Ponieważ odsyłacze pobierają tytuł z tematu docelowego, tekst łącza automatycznie dopasuje się do przetłumaczonego nagłówka w wynikach.
- Zmniejsza liczbę przeróbek: Jeśli tytuł powiązanego tematu ulegnie zmianie, nie musisz ręcznie aktualizować każdego łącza.
- Zminimalizuj błędy w tłumaczeniu: Hiperłącza z niestandardowym tekstem łącza mogą stać się nieaktualne, niedopasowane lub niespójne, jeśli nie będą uważnie śledzone.

Najlepsze praktyki lokalizacji:

- Używaj odsyłaczy krzyżowych podczas linkowania do treści wewnętrznej: Zawsze preferuj odsyłacze krzyżowe podczas wskazywania innego tematu lub nagłówka w ramach tego samego projektu. Dzięki temu link pozostanie ważny, nawet jeśli nazwa pliku lub nagłówek ulegną zmianie, a tekst będzie zgodny ze zlokalizowaną wersją.
- Unikaj zakodowanego tekstu w hiperłączach: Jeśli musisz użyć hiperłącza, unikaj pisania tekstu łącza ręcznie. Zamiast tego upewnij się, że jest on jasny i ogólny lub oznacz go do szczególnej uwagi w procesie tłumaczenia.
- Nie mieszaj odnośników zewnętrznych i hiperłączy w sposób niespójny: Wybierz jedną metodę na kontekst. Na przykład, jeśli konsekwentnie odwołujesz się do innych procedur lub sekcji w przewodniku, trzymaj się odsyłaczy krzyżowych w całym dokumencie, aby zachować jednolitą strukturę.

✓ Dobry przykład:
For more information, see <xref href="installing.md"/>.
✕Zły przykład:

For more information, clickhere.

Słowo „here” nie ma znaczenia poza kontekstem i należy go unikać we wszystkich dokumentach, zwłaszcza w tłumaczonych treściach.

Kiedy hiperłącza są odpowiednie

Istnieją pewne ważne przypadki użycia hiperłączy:

- Łączenie z zewnętrznymi witrynami internetowymi lub dokumentami zewnętrznymi
- Łącza e-mail (`mailto:`)
- Adresy URL, które nigdy się nie zmienią i nie wymagają tłumaczenia

W takich przypadkach pomocne jest:

- Używanie zmiennych dla zewnętrznych adresów URL w celu scentralizowania aktualizacji
- Używanie spójnego formatu dla wszystkich takich łączy

Umieszczanie tagów

Do czego się odnosi:

Sposób i miejsce umieszczania warunków, zmiennych lub fragmentów w tekście może drastycznie wpłynąć na możliwość tłumaczenia.

Najlepsze praktyki dotyczące lokalizacji:

- Stosuj tagi w logicznych punktach przerwania: Stosuj tagi warunkowe do całych akapitów lub elementów blokowych zamiast dzielić zdania. Tagi inline fragmentują treść i zakłócają działanie systemów pamięci tłumaczeniowej.
- Unikaj tagowania fragmentów zdania: Większość narzędzi do tłumaczenia działa na poziomie zdania. Jeśli zastosujesz tagi warunkowe do poszczególnych słów, mogą one wydawać się wyrwane z kontekstu dla tłumaczy.
- Utrzymuj kod w czystości: Unikaj stosowania wielu typów tagów (na przykład fragment + zmienna + warunek) w krótkim segmencie tekstu.

✓ Dobry przykład:

```
<p MadCap:conditions="Product_Pro">This feature is available in the Pro version only.</p>
```

Dlaczego jest dobry:

- Opisowa i precyzyjna nazwa
- Zastosowany do pojedynczego zdania w celu zapewnienia jasności
- Brak nakładających się lub złożonych warunków

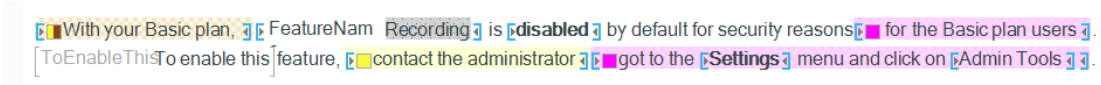
✕Słaby przykład:

```
<p>This feature is <span MadCap:conditions="Product_Pro">available</span> in the <Variable>ProductEdition</Variable>.</p>
```

Trudności w tłumaczeniu i utrzymaniu.

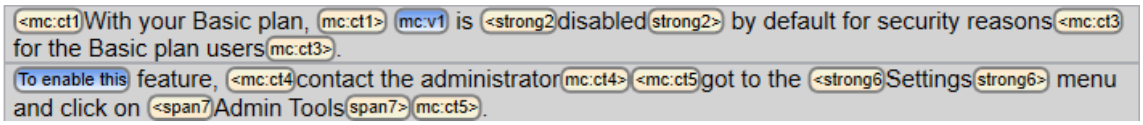
Punkt widzenia tłumacza

Umieszczanie wielu różnych tagów w jednym zdaniu lub nakładanie się tagów może zaciemniać ostateczny wynik, utrudniać pracę tłumaczom z treścią i zakłócać działanie systemów pamięci tłumaczeniowej. Tak może wyglądać tekst z nieprawidłowo zastosowanymi tagami w MadCap Flare:

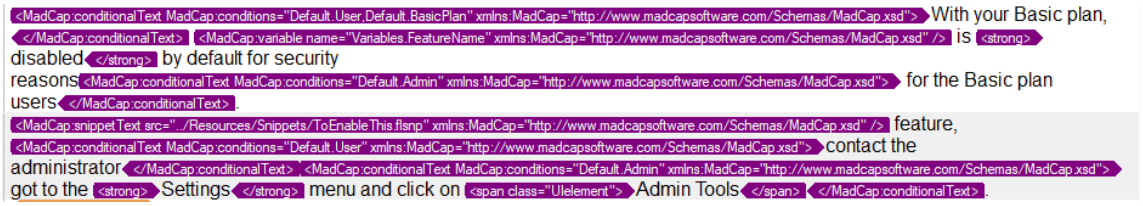


Tak tłumacz widzi ten tekst:

- MadCap Lingo



- SDL Trados Studio



Problemy:

- Nadmiar tagów powoduje bałagan wizualny, utrudniając zrozumienie struktury zdania i przewidywanie ostatecznych wyników.

Spłaszczanie snippetów i zmiennych rozwiązuje niektóre potencjalne problemy, ale nie wszystkie – inne tagi mogą nadal pozostać i przyczyniać się do zamieszania.

Formatowanie pod kątem lokalizacji

Podczas pisania treści pod kątem lokalizacji formatowanie jest często pomijane jako kwestia czysto wizualna. Jednak sposób formatowania treści – za pomocą arkuszy stylów, układów stron i struktury interfejsu użytkownika – może znacząco wpłynąć na koszt, złożoność i jakość procesu lokalizacji. Złe decyzje dotyczące formatowania mogą skutkować zakodowanym na stałe językiem, niedostępnymi stylami, uszkodzonymi układami w przetłumaczonych wynikach, a nawet koniecznością lokalizacji CSS lub ponownego formatowania wydruku dla każdego języka.

Ta sekcja koncentruje się na konkretnych kwestiach formatowania, które wpływają na gotowość do lokalizacji, zwłaszcza podczas pracy w ustrukturyzowanym środowisku, takim jak MadCap Flare.

Arkusze stylów kontra formatowanie inline

Jaki jest problem?

Formatowanie inline (np. ręczne ustawianie kolorów, grubości lub rozmiarów czcionek w edytorze WYSIWYG) może wydawać się wygodne, ale wprowadza poważne wyzwania lokalizacyjne.

✓ Najlepsze praktyki:

- Zawsze definiuj i stosuj style za pośrednictwem klas CSS lub globalnych arkuszy stylów, a nie inline.
- Nigdy nie używaj właściwości CSS `content` do wstrzykiwania widocznego lub przetłumaczalnego tekstu. Tekst osadzony w ten sposób jest ukryty przed przepływami pracy tłumaczeń (np. eksportami XLIFF).
- Przechowuj oddzielne arkusze stylów dla języków tylko wtedy, gdy jest to absolutnie konieczne (na przykład obsługa układu RTL), ale unikaj stylów specyficznych dla języka dla treści tekstowych.

✕Pułapki, których należy unikać:

- Dodawanie etykiet lub tytułów za pośrednictwem CSS, takich jak:

```
.warning::before { content: "Caution: "; }
```

Ta „Uwaga” nigdy nie zostanie przetłumaczona, chyba że sam arkusz stylów zostanie wyodrębniony i zlokalizowany ręcznie.

- Stosowanie formatowania takiego jak `<span style="font-weight:bold; color:red;">Important</span>` inline. Trudno sobie z tym poradzić na dużą skalę.

Lepsze podejście: Zdefiniuj `.important-note` w swoim CSS i zastosuj go jako klasę:

```
.important-note { font-weight: bold; color: red; }
```

Następnie po prostu zastosuj klasę w Flare.

Zezwalaj na rozszerzanie tekstu

Języki różnią się długością. Tłumaczenie z języka angielskiego na niemiecki, francuski lub rosyjski może spowodować, że tekst będzie dłuższy nawet o 30–50%. Brak miejsca na to w układzie może prowadzić do obcięcia, przepełnienia lub uszkodzenia elementów strony.

✓ Najlepsze praktyki:

- Projektuj z elastycznymi, responsywnymi kontenerami (nie polami tekstowymi o stałej szerokości).
- Zostaw dodatkową przestrzeń w makietach interfejsu użytkownika, tabelach i objaśnieniach.
- Unikaj wąskich kolumn lub ściśle ograniczonych układów w wynikach PDF.
- Rozważ podgląd treści z tekstem „Lorem ipsum”, który symuluje rozszerzenie (na przykład z duplikacją ciągu).

✕Pułapki, których należy unikać:

- Szerokości zakodowane na stałe w tabelach lub objaśnieniach, które ulegają uszkodzeniu podczas tłumaczenia.
- Zrzuty ekranu interfejsu użytkownika, które nie mogą pomieścić dłuższych etykiet tekstowych.
- Odstępy pionowe, które dokładnie pasują do języka angielskiego, ale ulegają uszkodzeniu w języku niemieckim lub fińskim.

> **Przykład:** Przycisk w języku angielskim oznaczony jako „Dalej” może stać się przyciskiem „Następny”; w języku polskim – psując układ interfejsu użytkownika, jeśli nie jest zaprojektowany z myślą o rozwijaniu.

Rozmiar papieru

Podczas drukowania (PDF, Word itd.) rozmiar papieru ma znaczenie – zarówno z perspektywy formatowania, jak i lokalizacji.

✓ Najlepsze praktyki:

- Używaj A4 jako domyślnego rozmiaru papieru, chyba że kierujesz swoją ofertę wyłącznie do odbiorców z Ameryki Północnej (gdzie standardem jest Letter).
- Jeśli generujesz wielojęzyczne wydruki, utwórz szablony, które dostosowują marginesy, odstępy i nagłówki zarówno do formatów A4, jak i Letter.
- Zwróć uwagę na to, jak rozszerzenie wpływa na paginację i podział stron w różnych językach.

✕Pułapki, których należy unikać:

- Zablokowanie układu do jednego rozmiaru papieru bez uwzględnienia ustawień regionalnych.
- Ustawianie zakodowanych na stałe odstępów między wierszami lub długości akapitów, których nie można dynamicznie dostosować za pomocą rozszerzenia.

> MadCap Flare umożliwia zdefiniowanie wielu układów stron. Używaj ich, aby dostosować wydruk do różnych języków i regionów.

Czcionki

Wybór czcionki nie jest tylko wyborem estetycznym – wpływa on na czytelność, obsługę znaków i stosowność kulturową w przetłumaczonych treściach.

✓ Najlepsze praktyki:

- Używaj czcionek zgodnych ze standardem Unicode, które obsługują szeroki zakres znaków (na przykład Arial Unicode MS, Noto Sans, Segoe UI).
- Wybieraj czcionki bezszeryfowe, aby zapewnić czytelność online, i rozważ czcionki szeryfowe do druku, jeśli jest to odpowiednie.
- Przetestuj, czy Twoja czcionka obsługuje znaki akcentowane, cyrylicę, znaki CJK (chińskie, japońskie, koreańskie) i pisma od prawej do lewej.

✕Pułapki, których należy unikać:

- Poleganie na niestandardowych lub dekoracyjnych czcionkach, którym brakuje zestawów znaków dla innych języków.
- Używanie czcionek, które nie są domyślnie instalowane w środowiskach zespołów tłumaczeniowych (może to powodować ponowne układanie tekstu lub brakujące znaki).
- Mieszanie typów czcionek w akapitach w sposób, który zaburza spójność po tłumaczeniu.

> Rodzina czcionek Noto firmy Google to popularny, przyjazny dla lokalizacji zestaw czcionek zaprojektowany specjalnie do obsługi wielu globalnych pism w sposób spójny.

Języki pisane od prawej do lewej (RTL) i ikony kierunkowe

Języki pisane od prawej do lewej, takie jak arabski, hebrajski, perski (farsi) i urdu, stwarzają szczególne wyzwania w zakresie formatowania – nie tylko w odniesieniu do kierunku tekstu, ale także wskazówek wizualnych, przepływu układu i ikonografii.

Rozważania dotyczące RTL:

- Kierunek układu musi być całkowicie odzwierciedlony – od wyrównania akapitu po menu nawigacyjne.
- Ikony sugerujące kierunek (na przykład strzałki, przyciski „Dalej”, do przodu/do tyłu) również muszą być odzwierciedlone, aby były zgodne z przepływem czytania.

✓ Najlepsze praktyki:

- Używaj obsługi RTL Flare w arkuszach stylów (`direction: rtl`) i układach stron.
- Utrzymuj odzwierciedlone wersje ikon, gdy wskazują one nawigację lub przepływ.
- Przetestuj wizualnie wynik w języku RTL, aby wychwycić niezgodności lub uszkodzone elementy.
- Upewnij się, że wszystkie znaczniki stylu i warunku są stosowane spójnie w całym odzwierciedlonym układzie.

✕Pułapki, których należy unikać:

- Używanie ikon strzałek „do przodu”, które wskazują w prawo → w interfejsie użytkownika w języku arabskim (zamiast tego powinny wskazywać ←).
- Stosowanie stylów wyrównanych do lewej do treści RTL lub zapominanie o odpowiednim wyrównaniu nagłówków i stopek.
- Osadzanie tekstu w obrazach – języki RTL mogą wymagać odbicia lustrzanego obrazu, co staje się trudne do wykonania, jeśli tekst jest w nich wbudowany.

> **Przykład – niezgodność ikon kierunkowych:**
>
> Ślad nawigacyjny, taki jak „Strona główna → Produkty → Szczegóły”, powinien stać się „Szczegóły ← Produkty ← Strona główna” w wynikach RTL, wraz z pasującym kierunkiem ikon.

Obrazy i zrzuty ekranu

Elementy wizualne, takie jak obrazy i zrzuty ekranu, są integralną częścią dokumentacji technicznej. Jednak bez starannego planowania mogą stać się znaczącymi przeszkodami w procesie lokalizacji. Wyzwania, takie jak osadzony tekst, stałe wymiary i brak skalowalności, mogą prowadzić do wzrostu kosztów i opóźnień. W tej sekcji opisano strategie optymalizacji obrazów i zrzutów ekranu pod kątem lokalizacji, wykorzystując narzędzia takie jak MadCap Capture i przestrzegając najlepszych praktyk branżowych.

Unikaj osadzania tekstu

Osadzanie tekstu bezpośrednio w obrazach (na przykład używanie narzędzi do projektowania graficznego do umieszczania tekstu na zrzutach ekranu) sprawia, że tekst staje się niedostępny dla narzędzi tłumaczeniowych. Ta praktyka wymaga ręcznego wyodrębniania i ponownego osadzania przetłumaczonego tekstu, co zwiększa ryzyko błędów i niespójności.

Najlepsze praktyki:

- Używaj objaśnień i podpisów: Zamiast osadzać tekst, używaj objaśnień lub podpisów, które są oddzielone od obrazu. MadCap Capture umożliwia tworzenie objaśnień, które są przechowywane w towarzyszącym pliku `.props` opartym na XML, ułatwiając łatwe tłumaczenie bez zmiany samego obrazu.
- Wykorzystuj grafikę wektorową: Jeśli to możliwe, używaj grafiki wektorowej do ilustracji. Formaty wektorowe umożliwiają oddzielenie tekstu od obrazów, umożliwiając proste aktualizacje tekstu bez modyfikowania grafiki.
- Zachowaj tekst w plikach źródłowych: Zachowaj całą zawartość tekstową w plikach źródłowych swojego projektu dokumentacji. To podejście zapewnia, że systemy pamięci tłumaczeniowej mogą uzyskać dostęp do tekstu i przetwarzać go wydajnie.

> **Przykład:**
> ✕Słaba praktyka: Zrzut ekranu z osadzonym tekstem w języku angielskim, takim jak „Kliknij tutaj, aby rozpocząć”, wymaga utworzenia nowego obrazu dla każdego języka.
> ✓ Poprawiona praktyka: Zrzut ekranu bez tekstu, któremu towarzyszy podpis lub wezwanie w dokumentacji, które brzmi „Kliknij przycisk „Start””

Zezwalaj na rozszerzanie tekstu

Tłumaczony tekst często zajmuje więcej miejsca niż tekst w języku oryginalnym. Na przykład niemieckie tłumaczenia mogą być nawet o 30% dłuższe od angielskich odpowiedników. Bez uwzględnienia tego rozszerzenia tekst może nachodzić na siebie lub wykraczać poza wyznaczone obszary na obrazach.

Najlepsze praktyki:

- Projektuj elastyczne układy: Podczas tworzenia objaśnień lub adnotacji upewnij się, że mogą się one rozszerzać lub zwężać w zależności od długości tekstu. MadCap Capture obsługuje automatyczne dostosowywanie rozmiaru pól tekstowych, które dynamicznie dostosowuje rozmiar do zawartości.
- Testuj z pseudolokalizacją: Wdrażaj techniki pseudolokalizacji, aby symulować rozszerzanie tekstu i identyfikować potencjalne problemy z układem przed faktycznym tłumaczeniem.
- Unikaj kontenerów o stałym rozmiarze: Powstrzymaj się od używania kontenerów tekstowych o stałym rozmiarze w obrazach. Zamiast tego zezwól na dynamiczną zmianę rozmiaru, aby dostosować się do różnych długości tekstu.

> **Przykład:**
> ✕Zła praktyka: Pole objaśnienia o stałym rozmiarze, które pasuje tylko do angielskiego tekstu „End”, może nie pomieścić polskiego tłumaczenia „Zakończ”, co prowadzi do przepełnienia tekstu.
> ✓ Ulepszona praktyka: Dynamicznie zmieniający rozmiar pola objaśnienia, które dostosowuje się do przetłumaczonego tekstu, zachowując czytelność i estetykę.

Unikaj zakodowanych na stałe wymiarów

Zakodowanie na stałe wymiarów obrazu może prowadzić do problemów z wyświetlaniem na różnych urządzeniach i w różnych językach. Stałe rozmiary mogą nie uwzględniać rozszerzenia tekstu lub różnych rozdzielczości ekranu, co skutkuje zniekształconymi lub obciętymi obrazami.

Najlepsze praktyki:

- Używaj względnego określania rozmiaru: Definiuj rozmiary obrazu, używając jednostek względnych (na przykład procentów) zamiast stałych pikseli. Takie podejście zapewnia odpowiednie skalowanie obrazów na różnych urządzeniach i w różnych układach.
- Implementuj responsywny projekt: Projektuj obrazy i układy, które dostosowują się do różnych rozmiarów i orientacji ekranu, zwiększając dostępność i komfort użytkownika.
- Testuj na różnych wyjściach: Podgląd obrazów w różnych formatach wyjściowych (HTML5, PDF), aby zapewnić spójny wygląd i funkcjonalność.

> **Przykład:**
> ✕Zła praktyka: Obraz ustawiony na stałą szerokość 600 pikseli może nie wyświetlać się prawidłowo na mniejszych ekranach lub nie mieścić dłuższego przetłumaczonego tekstu.
> ✓ Ulepszona praktyka: Obraz ustawiony tak, aby zajmował 100% szerokości kontenera, co umożliwia jego odpowiednie skalowanie w zależności od urządzenia wyświetlającego.

Wykorzystaj pliki warstwowe i MadCap Capture

Zarządzanie wieloma wersjami obrazów dla różnych języków może być czasochłonne i podatne na błędy, szczególnie w przypadku osadzonego tekstu i adnotacji.

Najlepsze praktyki:

- Wdrażanie MadCap Capture: Używaj MadCap Capture do tworzenia obrazów warstwowych, w których adnotacje tekstowe są przechowywane oddzielnie od obrazu w plikach `.props`. To rozdzielenie upraszcza proces tłumaczenia i utrzymuje spójność między językami.
- Utrzymuj pliki źródłowe: Utrzymuj oryginalne, edytowalne pliki obrazów uporządkowane i dostępne. Ta praktyka ułatwia aktualizacje i modyfikacje bez konieczności ponownego tworzenia obrazów od podstaw.
- Integracja z przepływami pracy nad tłumaczeniem: Upewnij się, że warstwy tekstowe na obrazach są uwzględnione w przepływie pracy nad tłumaczeniem, umożliwiając tłumaczom dostęp do tekstu i jego modyfikację bez zmiany samego obrazu.

> **Przykład:**
> ✕Zła praktyka: Tworzenie oddzielnych obrazów dla każdego języka, osadzanie przetłumaczonego tekstu bezpośrednio w każdej wersji.
> ✓ Ulepszona praktyka: Używanie pojedynczego obrazu z warstwami tekstowymi zarządzanymi przez MadCap Capture, co umożliwia wydajne aktualizacje i spójność we wszystkich wersjach językowych.

Wybierz metodę tłumaczenia

Podczas przygotowywania projektu Flare do lokalizacji ważne jest, aby wybrać odpowiednią metodę tłumaczenia. Każda metoda ma określone zalety i wady w zależności od rozmiaru projektu, złożoności, harmonogramu i dostępnych zasobów. Tutaj przyjrzymy się najczęstszym podejściom do tłumaczenia projektów MadCap Flare: przy użyciu plików wyjściowych, przy użyciu XLIFF i przy użyciu MadCap Lingo.

Szybkie porównanie

Metoda	Najlepsza dla	Główna zaleta	Główna wada
"Pliki wyjściowe" na następnej stronie	Małe, proste projekty	Łatwe do wysłania	Ręczne, podatne na błędy tłumaczenie i ponowna integracja
"XLIFF" na stronie131	Profesjonalna lokalizacja	Zachowująca strukturę, wydajna	Wymaga konfiguracji, bardziej techniczna
"MadCap Lingo" na stronie133	Zintegrowany wewnętrzny przepływ pracy	Pełna zgodność z Flare, wbudowana kontrola jakości	Osobne narzędzie, wymagana licencja

Pliki wyjściowe

Najwyraźniej jedną z najprostszych metod lokalizacji jest generowanie plików wyjściowych (takich jak dokumenty Word, HTML lub PDF) z Flare i wysyłanie ich do tłumacza.

Jak to działa

- Generujesz sfinalizowany, czytelny dla człowieka wynik (na przykład Word .docx).
- Tłumacz edytuje tekst bezpośrednio w pliku wyjściowym.
- Ponownie importujesz przetłumaczony dokument do MadCap Flare lub ręcznie aktualizujesz projekt.

Zalety

- Prostota: Łatwy do zrozumienia dla użytkowników nietechnicznych i tłumaczy niezaznajomionych z Flare.
- Natychmiastowa użyteczność: Tłumaczenie można rozpocząć, gdy tylko dane wyjściowe będą gotowe, bez potrzeby korzystania ze specjalistycznych narzędzi do tłumaczenia.

Wady

- Ręczne przerabianie: Przetłumaczony plik często musi być ręcznie kopiowany z powrotem do tematów Flare lub ponownie tworzony, co jest czasochłonne i podatne na błędy.
- Utrata struktury: Wyniki nie są ustrukturyzowane jak oryginalny projekt Flare. Tracisz fragmenty kodu, zmienne i metadane. Należy unikać stosowania warunków, ponieważ nie są one rozpoznawane i nie zostaną prawidłowo zastosowane.
- Słaba skalowalność: Nadaje się tylko do bardzo małych, statycznych projektów z minimalnymi aktualizacjami.

Najlepsze przypadki użycia

- Bardzo małe projekty lub pojedyncze pliki bez tekstu warunkowego.
- Jednorazowe lub okazjonalne zadania tłumaczeniowe bez zaplanowanych aktualizacji.
- Odbiorcy nietechniczni, dla których tłumaczenie musi być szybkie i proste.

> Zaleca się, aby zawsze tłumaczyć pliki źródłowe. Tłumacząc projekt źródłowy (a nie pliki wyjściowe), zwiększasz integralność projektu, unikasz potencjalnych problemów z przepływem pracy i konserwacją oraz zmniejszasz ogólne koszty tłumaczenia.

XLIFF

XLIFF (XML Localization Interchange File Format) to standardowy w branży format pliku zaprojektowany specjalnie do wymiany przetłumaczonych treści między twórcami treści a zespołami lokalizacyjnymi.

Jak to działa

- Eksportujesz zawartość z Lingo jako pliki XLIFF.
- LSP lub tłumacz pracuje bezpośrednio z plikami XLIFF przy użyciu narzędzia do tłumaczenia wspomaganego komputerowo (CAT).
- Po zakończeniu tłumaczenia importujesz pliki XLIFF z powrotem do Lingo, a następnie do Flare.

Zalety

- Ustrukturyzowane dane: Zachowuje strukturę tekstu, metadane i segmentację, co znacznie ułatwia ponowną integrację.
- Wydajność aktualizacji: Obsługuje pamięć tłumaczeniową (TM), umożliwiając ponowne wykorzystanie wcześniej przetłumaczonych segmentów.
- Lepsza kontrola jakości: Narzędzia CAT umożliwiają sprawdzanie spójności, dopasowywanie glosariuszy i wykrywanie błędów podczas tłumaczenia.
- Przyjazny dla automatyzacji: Przepływy pracy XLIFF dobrze integrują się z zautomatyzowanymi potokami lokalizacyjnymi.

Wady

- Konfiguracja eksportu/importu: Musisz poprawnie skonfigurować eksport (ręcznie wybrać pliki, wykluczyć nieprzetłumaczalną zawartość, wyraźnie uporządkować tematy).
- Wrażliwość na błędy: Jeśli nie zostaną odpowiednio obsłużone, niezgodności struktury lub błędy tagów mogą przerwać ponowne importowanie do Flare.

Najlepsze przypadki użycia

- Duże lub stale aktualizowane projekty.
- Projekty wymagające bieżącego zarządzania pamięcią tłumaczeniową.
- Witryny wielojęzyczne, w których dokładność i spójność mają kluczowe znaczenie.

> W MadCap Lingo możesz eksportować pliki XLIFF dla tematów, fragmentów i innych zasobów. Gdy tłumacze zwrócą ukończone pliki XLIFF, możesz je ponownie zaimportować, a Flare automatycznie zaktualizuje zawartość, nie naruszając struktury projektu.

MadCap Lingo

MadCap Lingo to narzędzie do zarządzania tłumaczeniami opracowane przez MadCap Software specjalnie do pracy z projektami Flare.

Jak to działa

- Tworzysz projekt Lingo na podstawie swojego projektu Flare.
- Lingo zajmuje się ekstrakcją, tłumaczeniem, przeglądem i ponowną integracją treści.
- Obsługuje TM, glosariusze, integrację tłumaczeń maszynowych i kontrole jakości.

Zalety

- Ścisła integracja z Flare: Bezpośrednio rozumie struktury Flare – tematy, fragmenty, zmienne, warunki, a nawet mikrotreści.
- Kompleksowy przepływ pracy: Zarządzanie projektem, tłumaczenie i QA są wbudowane w Lingo.
- Zachowuje strukturę: Brak ryzyka utraty fragmentów, warunków lub złożonych linków.
- Ułatwia wewnętrzne tłumaczenie: Umożliwia firmom zatrudniającym wewnętrznych tłumaczy zarządzanie tłumaczeniem bez konieczności korzystania z zewnętrznych narzędzi CAT.
- Obsługa pamięci tłumaczeniowej i glosariusza: Automatyczne ponowne wykorzystywanie poprzednich tłumaczeń w różnych projektach.

Wady

- Krzywa uczenia: Wymaga nauki interfejsu i funkcjonalności Lingo.
- Koszt licencji: Wymaga oddzielnej licencji od Flare.
- Ograniczona adopcja LSP: Nie wszyscy LSP znają Lingo, więc jeśli zlecasz pracę na zewnątrz, możesz potrzebować zamiast tego wyeksportować XLIFF.

Najlepsze przypadki użycia

- Organizacje, które chcą mieć pełną kontrolę nad procesem tłumaczenia.
- Projekty z dużą ilością ponownego użycia (zmienne, fragmenty, warunki), których ręczne zarządzanie byłoby uciążliwe.
- Firmy z wewnętrznymi autorami technicznymi, którzy również tłumaczą lub zarządzają lokalizacją.

> Możesz utworzyć projekt Lingo na podstawie swojego projektu Flare. Po zakończeniu tłumaczenia w Lingo przez wewnętrznego tłumacza (lub zewnętrznego lingwistę), po prostu synchronizujesz przetłumaczoną treść z powrotem do Flare przy minimalnym wysiłku ręcznym.

Generowanie projektów do tłumaczenia

Zanim praca nad tłumaczeniem będzie mogła się rozpocząć, projekt Flare musi zostać przygotowany i poprawnie wyeksportowany. Wybrana metoda będzie zależeć od narzędzi, których używasz Ty i Twój partner lokalizacyjny (LSP).

Ta sekcja przedstawia trzy kluczowe procedury eksportowania projektów do tłumaczenia:

- Eksportowanie bez MadCap Lingo (przy użyciu plików wyjściowych)
- Eksportowanie z MadCap Lingo (przy użyciu plików XLIFF)
- Eksportowanie do bezpośredniego tłumaczenia w MadCap Lingo

Eksportowanie plików wyjściowych

Ta metoda wykorzystuje czytelne dla człowieka pliki wyjściowe (na przykład dokumenty HTML lub Word) eksportowane z Flare. Jest to podstawowe podejście, gdy tłumacze nie używają narzędzi CAT lub gdy Lingo nie jest dostępne.

1. Eksportuj swój projekt Flare w odpowiednim formacie docelowym (Word lub HTML) (aby zobaczyć tę procedurę, przejdź do sekcji "Eksportowanie projektu Flare" na stronie138).
2. Wypakuj pliki ręcznie z folderu wyjściowego projektu.
3. Przekaż pliki do LSP wraz z:
    - Wszelkimi niezbędnymi instrukcjami dotyczącymi formatowania.
    - Przewodnikiem po stylach i słownikiem, jeśli są dostępne.
4. Po przetłumaczeniu:
    - Tłumacz zwraca przetłumaczone pliki Word lub HTML.
    - Ręcznie kopiujesz i wklejasz przetłumaczoną treść z powrotem do swojego projektu Flare.
    - Ponownie zastosuj fragmenty, zmienne, warunki w razie potrzeby (jeśli zostały spłaszczone podczas eksportu).

Zanim wyeksportujesz pliki z Lingo – przygotowanie

Aby pomyślnie wyeksportować pliki do tłumaczenia w MadCap Lingo lub narzędziu CAT innej firmy, wymagana jest pewna konfiguracja. Te kroki przygotowawcze zapewniają, że zarówno projekt Flare, jak i środowisko Lingo są poprawnie skonfigurowane, aby zapewnić płynny przepływ pracy nad tłumaczeniem.

Eksportowanie projektu Flare

Możesz wyeksportować cały projekt Flare lub jego części, aby użyć go do tłumaczenia. Jeśli potrzebujesz przetłumaczyć tylko część projektu nadrzędnego, nie chcesz wysyłać tłumaczowi wszystkich plików, ale raczej mniejszą wersję projektu zawierającą tylko pliki wymagające tłumaczenia.

Możesz wyeksportować projekt, korzystając z Kreatora eksportu projektu, który Cię poprowadzi. Alternatywnie możesz dodać plik eksportu do swojego projektu i użyć Export Project File Editor.

> UWAGA: Ta procedura opisuje podstawową konfigurację eksportu. Aby uzyskać kompleksowy przewodnik dotyczący eksportowania projektów Flare, odwiedź oficjalną stronę MadCap Flare: Eksportowanie projektów

Jak wyeksportować projekt za pomocą Kreatora eksportu projektu

1. Otwórz projekt.
2. Wybierz Project → Export Project. Otworzy się Kreator eksportu projektu.
3. W polu New Project Name wprowadź nazwę eksportowanego projektu.

   > (Opcjonalnie) Pole Ścieżka nowego projektu zostanie automatycznie wypełnione domyślną lokalizacją (`Documents\My Project Exports`). Jeśli chcesz wyeksportować projekt do innej lokalizacji, kliknij wielokropek przeglądania i wybierz żądany folder.

4. Na liście rozwijanej Export From wybierz jedną z opcji.
   - Entire Project: Ta opcja tworzy kopię całego projektu, w tym wszystkich folderów, podfolderów i plików.
   - Using Target: Ta opcja używa tego samego przepływu pracy, co ten używany do generowania celu. Po wybraniu konkretnego celu te same pliki i zawartość, które byłyby uwzględnione w wygenerowanym wyjściu, są uwzględniane w wyeksportowanym projekcie.
   - Using Conditions: Ta opcja eksportuje tylko pliki i zawartość, na które mają wpływ znaczniki warunków, które Flare ma uwzględnić lub wykluczyć.

- Using File Tags: Ta opcja eksportuje tylko pliki, na które mają wpływ znaczniki plików, które Flare ma uwzględnić lub wykluczyć. Ta metoda może być szczególnie przydatna do celów tłumaczeniowych, eksportując tylko pliki oznaczone określonym statusem znacznika pliku (na przykład Gotowe do tłumaczenia).
- Select Files: Ta opcja eksportuje tylko określone pliki, które wybierzesz. Możesz wybrać dowolne pliki zapisane w Eksploratorze zawartości i Organizatorze projektu źródłowego.

5. Na liście rozwijanej wybierz jedną z opcji.
    - Zip File: Ta opcja pakuje pliki projektu do pojedynczego pliku ZIP z rozszerzeniem .flprjzip i umieszcza go w wybranej lokalizacji. Domyślna lokalizacja to `Documents\My Project Exports`.
    - Project Files: Ta opcja po prostu eksportuje pliki projektu do wybranej lokalizacji. Domyślna lokalizacja to `Documents\My Project Exports`.
    - Project Template: Ta opcja eksportuje pliki projektu do folderu szablonów (na przykład `Documents\My Templates\Projects`). Po umieszczeniu w tej lokalizacji pliki projektu stają się dostępne jako wybór szablonu podczas tworzenia nowego projektu.
6. Kliknij przycisk Next.

    (Opcjonalnie) Możesz wybrać jedną lub obie z następujących opcji:

    - Convert variables to text: Wybierz tę opcję, jeśli chcesz, aby wszystkie odpowiednie zmienne zostały przekonwertowane na tekst.
    - Convert snippets to text: Wybierz tę opcję, jeśli chcesz, aby wszystkie odpowiednie fragmenty kodu zostały przekonwertowane na tekst.
7. Kliknij Finish.

Tworzenie projektu Lingo

Możesz utworzyć nowy projekt tłumaczenia Lingo za pomocą Kreatora nowego projektu. Ten kreator umożliwia wybór spośród wielu różnych rodzajów plików (na przykład projektów MadCap Flare, projektów Doc-To-Help, dokumentów Microsoft Word, Adobe FrameMaker, Adobe InDesign, plików XML, plików DITA), a także określenie jednego lub więcej języków docelowych. Możesz również dodać całe foldery do projektu, a następnie wybrać typy plików, które chcesz przetłumaczyć.

> UWAGA: Ta procedura opisuje podstawową konfigurację eksportu. Aby uzyskać kompleksowy przewodnik dotyczący eksportowania projektów Flare, odwiedź oficjalną stronę internetową MadCap Lingo: Tworzenie projektów

Jak utworzyć nowy projekt

1. Wykonaj jedną z następujących czynności, w zależności od używanej części interfejsu użytkownika:
   - Wstążka: Select File → New Project.
   - Skrót klawiaturowy: Naciśnij CTRL+SHIFT+N.

   Otworzy się Kreator nowego projektu.
2. W polu Name wpisz odpowiednią nazwę swojego projektu.
3. Domyślnie ścieżka do folderu `Documents\My Translated Projects` na dysku twardym jest wprowadzana w polu Folder. Przejdź do następnego kroku, chyba że chcesz umieścić pliki projektu w innym folderze.

   > (Opcjonalnie) W polu Domain możesz wpisać kategorię tematu dla swojego projektu lub wybrać ją z listy rozwijanej. Metadane domeny pojawiają się w całym projekcie, np. gdy wybierzesz sugestię w pamięci tłumaczeniowej.
   >
   > (Opcjonalnie) W polu Client możesz wpisać nazwę klienta, dla którego tworzysz projekt tłumaczenia. Metadane klienta pojawiają się w całym projekcie, np. gdy wybierzesz sugestię w pamięci tłumaczeniowej.
4. W liście rozwijanej Source Language wybierz oryginalny język używany w tłumaczonym projekcie. Upewnij się, że ten język dokładnie odpowiada językowi źródłowemu. Na przykład możesz tutaj wybrać English (en), ale pliki źródłowe są w rzeczywistości w języku English United States (en-us). Dlatego w tym polu należy wybrać opcję English (United States).

5. W siatce Target Languages wybierz język(i), których chcesz użyć do tłumaczenia i kliknij ikonę strzałki w prawo, aby przenieść je do siatki Selected Languages. Jeśli chcesz usunąć język z siatki Selected Languages, kliknij ikonę strzałki w lewo.

   (Opcjonalnie) Jeśli chcesz powiązać projekt Lingo z dostawcą kontroli źródła, wybierz opcję Bind to source control.

6. Kliknij przycisk Next.

   (Opcjonalnie) Jeśli wybrano opcję Bind to Source Control, kliknij Bind Project. Zostanie wyświetlone okno dialogowe Bind Project (Powiąż projekt). Wykonaj wszystkie odpowiednie kroki i po zakończeniu kliknij przycisk Next.

7. Dodaj pliki lub foldery do projektu Lingo lokalnie lub z kontroli źródła. Jeśli wybierzesz folder, pliki w nim zawarte zostaną dodane.

   Aby dodać pliki do projektu Lingo

   a. Kliknij przycisk Add File.

   b. W otwartym oknie dialogowym przejdź do projektu lub plików, które chcesz przetłumaczyć. Może być konieczne kliknięcie listy rozwijanej file type, aby wyświetlić i wybrać odpowiedni rodzaj plików, które chcesz dodać.

   c. Jeśli chcesz wybrać tylko jeden plik, możesz go kliknąć dwukrotnie. Jeśli chcesz wybrać wiele plików (na przykład wiele dokumentów Word) w tym samym folderze, przytrzymaj klawisz SHIFT lub CTRL i wybierz zakres lub pojedyncze pliki. Następnie kliknij Open.

8. Kliknij przycisk Next.

   (Opcjonalnie) Jeśli tworzysz projekt przy użyciu plików źródłowych MadCap Flare lub DITA, wybierz dodatkowe ustawienia projektu (spłaszcz zmienne, spłaszcz fragmenty kodu, utwórz skórkę języka itd.).

9. Kliknij przycisk Next.

   (Opcjonalnie) Dostępne bazy danych pamięci tłumaczeniowej (TM) są wyświetlane w siatce i możesz wybrać te, których chcesz użyć w bieżącym projekcie.

10. Kliknij przycisk Next.

    (Opcjonalnie) Dostępne bazy terminologiczne są wyświetlane w siatce i możesz wybrać te, których chcesz użyć w bieżącym projekcie.

11. Kliknij Finish.

Podczas tworzenia nowego projektu Lingo automatycznie generowany jest plik z rozszerzeniem .liprj (na przykład `My Project Polish.liprj`).

Eksportowanie plików XLIFF

XLIFF (XML Localization Interchange File Format) to standard wymiany danych lokalizacyjnych. Chociaż Flare nie generuje plików XLIFF bezpośrednio, możesz wyeksportować zawartość Flare do MadCap Lingo, a następnie utworzyć pliki XLIFF do tłumaczenia.

1. Wyeksportuj swój projekt Flare (aby zobaczyć tę procedurę, przejdź do sekcji "Eksportowanie projektu Flare" na stronie138).
2. Utwórz projekt Lingo z Flare (aby zobaczyć tę procedurę, przejdź do sekcji "Tworzenie projektu Lingo" na stronie140).
3. Wyeksportuj pakiet XLIFF, aby wysłać go do swojego LSP:
    a. Wybierz View → File List.
    b. Wybierz pliki, które mają zostać uwzględnione w pakiecie.
    c. Na lokalnym pasku narzędzi kliknij przycisk Create Bundle.
    d. W obszarze Bundle Settings wybierz XLIFF Files Bundle i zaznacz pliki projektu, które mają zostać uwzględnione w pakiecie.
    e. Dołącz pamięci tłumaczeń projektu, jeśli to konieczne.
    f. Dołącz bazy terminologiczne projektu, jeśli to konieczne.
    g. Kliknij OK. Zostaną wyświetlone statystyki dla projektu.
    h. Kliknij Close. Otworzy się podfolder (w folderze projektu) o nazwie Eksportowanie. Ten podfolder zawiera plik ZIP. Ten plik zawiera pliki, z którymi musi pracować tłumacz.
4. Dostarcz pliki XLIFF do LSP wraz z przewodnikiem stylu, jeśli jest dostępny.
5. Po zakończeniu i sprawdzeniu tłumaczenia zaimportuj pliki XLIFF z powrotem do projektu Lingo.
6. Wyeksportuj projekt Lingo do projektu Flare w języku docelowym.

Eksportowanie do tłumaczenia w Lingo

Jeśli Twój obieg tłumaczeń opiera się na Twojej organizacji lub jeśli Twój LSP używa MadCap Lingo, możesz eksportować i tłumaczyć całkowicie w środowisku Lingo bez potrzeby używania oddzielnych plików.

1. Eksportuj swój projekt Flare (aby zobaczyć tę procedurę, przejdź do sekcji"Eksportowanie projektu Flare" na stronie138).
2. Utwórz projekt Lingo z Flare (aby zobaczyć tę procedurę, przejdź do sekcji "Tworzenie projektu Lingo" na stronie140).
3. Przetłumacz bezpośrednio w Lingo lub wyeksportuj pakiet LIPRJZIP, aby wysłać go do swojego LSP. Aby utworzyć pakiet LIPRJZIP:
    a. Wybierz View → File List.
    b. Wybierz pliki, które mają zostać uwzględnione w pakiecie.
    c. Na lokalnym pasku narzędzi kliknij przycisk Create Bundle.
    d. W obszarze Bundle Settings wybierz Lingo Project Bundle i zaznacz pliki projektu, które mają zostać uwzględnione w pakiecie.
    e. Dołącz pamięci tłumaczeń projektu, jeśli to konieczne.
    f. Dołącz bazy terminologiczne projektu, jeśli to konieczne.
    g. Kliknij OK. Zostaną wyświetlone statystyki dla projektu.
    h. Kliknij Close. Otworzy się podfolder (w folderze projektu) o nazwie Eksportowanie. Ten podfolder zawiera plik LIPRJZIP. Ten plik zawiera pliki, z którymi musi pracować tłumacz.
4. Wyślij plik LIPRJZIP do tłumacza wraz z przewodnikiem po stylach, jeśli jest dostępny.
5. Po zakończeniu tłumaczenia i sprawdzeniu zsynchronizuj przetłumaczoną treść z powrotem do Flare.