

Developer Guide | PUBLIC
Document Version: 1.0 – 2025-06-06

Guidelines for Developing Multilingual Applications

© 2025 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Guidelines for Developing Multilingual Applications. 3

1.1 The Basics. 3

1.2 Technical Translatability. 4

1.3 Truncated Text, Length Restrictions, and Abbreviations. 4

1.4 Variables in Messages and Other UI Texts. 6

1.5 English: Capitalization (Sentence Style and Title Case) of UI Elements. 9

1.6 Context Information for Translators. 10

1.7 Wording and Formulations. 12

1.8 Typography and Punctuation Marks. 13

1 Guidelines for Developing Multilingual Applications

User interfaces (UIs) are translated into many languages. This documentation contains some guidelines and best practices that developers can follow to ensure that UI texts can be translated into other languages. People in nondeveloper roles can use the best practices to help determine whether projects are being developed in accordance with multilingual development principles.

UI texts comprise all kinds of labels and titles, such as those for fields, buttons, menus, and columns, as well as messages and embedded help like tooltips or permanent on-screen explanations.

1.1 The Basics

Create a framework for ensuring good quality texts in your applications right at the beginning of a project.

The following points can help you to organize this framework:

- Even when you are planning a project, involve user assistance developers or technical writers and translators.
  Invite them to mockup sessions and UI reviews. Ask them to review the UI texts, including messages. If required, arrange for someone to language edit all texts. You can rely on their expertise for terminology and comprehensibility. They rely on your subject matter expertise and willingness to ensure translatability.

  > ⚠ Caution
  >
  > UI text reviews and language editing are crucial for the user experience and the translation process. By increasing the quality of the source texts, you can save lots of time, money, and nerves. If you don't want to be swamped by translation queries from all those languages, establish language editing processes. One approach is for language editors to work directly in the source environment, such as an ABAP system or GitHub.

- Externalize text strings. Do not use hard-coded text.
  Hard-coded text won't be translated because it is not detected by translation tools.
- Use agreed terminology and product names. Do not use synonyms – they cause confusion.
  If you plan to change terminology or formulations for common UI texts, please contact your user assistance developers and translators upfront.
- Create texts only in the original language of the object or project.
  Otherwise, your customers will get screens with text in the wrong language, which they find at best annoying and at worst completely confusing. If any cases of wrong language usage slip through the net, you'll have to correct these cases after the event, which you'll find annoying and time-consuming. Translation tools and memories work only if a text with a given source language really contains text in that language.
- Before mass translation starts, remove test and dummy text.
  By removing these texts, you help to reduce translation costs and time. You will also receive fewer translation queries.

1.2 Technical Translatability

To be able to make the most of good translation resources, it must be possible to translate the texts in your applications UIs.

In a nutshell, making text translatable covers the following points:

- Even if you don't plan to translate your UIs into other languages at the moment, make the product translatable. By doing so, you can translate your product whenever market requirements or other demands require you to do so.
- Extract translatable texts and store separately from the source code. For example, for Java applications, you store translatable texts in properties files.
- Consider which texts in your product or application require translation:
  - All texts shown to a user during normal processing (including common messages) almost certainly need to be translated
  - Technical texts, runtime errors, or log entries for technical logs **might not** need to be translated
- Consider the format that you use for files that contain externalized texts - can your translation tool of choice, such as SAP Translation Hub, process the files?
- Provide context information wherever possible; translators depend heavily on context information to be able to translate texts correctly.

1.3 Truncated Text, Length Restrictions, and Abbreviations

To ensure that abbreviations are kept to an absolute minimum, you can take appropriate measures during the development phase.

> ⓘ Note
>
> Truncated and cryptically abbreviated text make screens unusable, not just in the original language. If English is your source language, you will almost certainly need more space on the UI for other languages. So, it's important to know how to deal with length restrictions and space on the UI. The following screenshot is an example of a screen with truncated field labels, even though there's lots of white space.



To ensure a minimum of abbreviations and no truncated texts on your UIs, see the following points:

- Do **not** set text length restrictions of 1–3 characters.
  Texts with such length restrictions cannot be translated easily into double-byte languages, such as Chinese or Japanese.
  Exception: There can be exceptions. For example, the 1-letter or 2-letter abbreviations of weekdays in the calendar can be translated into double-byte languages because these languages have equivalent short forms.
- Eliminate cryptic, nonstandard abbreviations.
  Abbreviations are difficult or even impossible to understand and translate. Try removing or increasing text length restrictions, adapting the screen areas, or rewording the text.

  > Example
  >
  > ✔ Project Forecast [or:] Forecast
  >
  > ✘ Proj. Fcst.
  >
  > ✔ Payment Currency [or:] Currency
  >
  > ✘ PyC

  Exception: You can, of course, use common or industry-standard abbreviations, such as “ID”, “CPU”, “ABAP”, or “VPN”. They are more common than their full forms, but their acceptance might depend on the target group.
- Text string length and space on the UI are two different things.
  Both cause truncation or force you to use abbreviations. Here's what to do:
  - Do you need to restrict the length at all?
    If you don't need to restrict the length, don't. This applies, for example, to HTML5 or Java developments. If you don't impose length restrictions, translators don't have to come up with abbreviations in the target languages.
    Exception: Restrict the length manually **only if** space on the screen is absolutely critical or there is no automatic mechanism for handling longer text.
  - If the string length is restricted (you have to define it or it is predefined in the system), allow the maximum number of characters for the text type. Use the table below as a guide for the recommended string length.
  - Are UI texts truncated in other logon languages?
    Even if the text string length is not restricted, the UI will not automatically look great in every language. You might still need to assign more space to certain screen areas.
- Do something proactive against truncated text, cryptic abbreviations, and hard-coded text.
  If you do have to restrict the length of fields on the UI, see the following table for recommendations based on English as the source language:

  Recommendations for Minimum Field Length

Length of Source Text (Characters)	Minimum Field Length Required for Localization (Characters)
1 <= 4	10
5	14

Length of Source Text (Characters)	Minimum Field Length Required for Localization (Characters)
6	16
7	18
8 <= 10	20
11	22
12	24
13 <= 15	26
16	28
17 <= 20	32
21 <= 80	Number of characters in source text + 50%
> 80	Number of characters in source text + 30%

1.4 Variables in Messages and Other UI Texts

When you use variables in texts, ensure that it is possible to translate these texts and that it is possible for translation tools and translators to make sense of the texts that you create using variables.

Do not use variables to represent translatable words (nouns, verbs, adjectives, ...).

Why? Because this makes the text untranslatable. There'll be issues such as incomprehensible sentences with confusing word order or incorrect verb/noun/adjective agreement in the target languages. And for you as a developer, this means lots of requests from translators for you to sort out these sentences that contain variables for translatable words.

Therefore, stick to this:

- Use variables **only for untranslatable items** such as object IDs, values, or technical names.
- Address the variables uniquely by numbering them or by using descriptive names. This helps translators to place each variable in the correct position in the sentence.

> ✤ Example
>
> [Note: `<variable 1>` is a neutral placeholder for notations such as `&1` in ABAP or {0} in i18n properties files.]
>
> ✖ Do you want to `<variable>` the `<variable>`? [Note: **Untranslatable!** No clue what the variables contain. Also, the English word order and grammatical structure will not work in other languages. For example, many languages rely on inflectional endings. To choose the correct ending, a translator needs to know the complete sentence and its words. However, this is impossible because the sentence and the variables are translated independently of each other. Also, in many languages, the translation of the article “the” is likely to result in a messy mix of slashes, such as German “den/die/das” or French “l'/la/le/les”.]

✔ Add device `<variable 1 (here: an ID)>` to group `<variable 2 (here: an ID)>`. [Note: Good! The variables refer to untranslatable items (in this case: IDs) instead of translatable words. The variables are part of the text string and they are preceded by descriptive words.]

✔ Do you want to approve shopping cart `<variable 1 (here: an ID)>`? [Note: Everything fine. Compare it to the incorrect examples.]

✘ Do you want to approve shopping cart ? [Note: The variable is invisible. Looks like it's generated into the text at runtime. Translators cannot move it to the right position in the target language.]

Exception: In rare cases, you might not be able to avoid a variable with translatable text – for example, if you want to point the user to a Web site. In such a case, separate the variable from all the other words in the sentence. This is the only way to make the sentence translatable for all types of target languages.

Example

[Note: `<variable 1>` is the version number. `<variable 2>` is the text "SAP Store" with an underlying URL.]

✔ Download SAP Lumira `<variable 1>` from this site: `<variable 2>`

✘ Download SAP Lumira `<variable 1>` from `<variable 2>`. [Note: Won't work with languages that rely on inflectional endings.]

Do not concatenate words to form sentences even if it is tempting.

Different languages have a different word order and grammar. In the target languages, concatenated text results in incorrect, incomprehensible, or even nonsensical sentences. The translator doesn't stand a chance of providing a good translation. Users are likely to be annoyed.

Example

[Note: Write down complete sentences, without variables.]

✔ Do you want to create the order?

✔ Do you want to change the order?

✔ Do you want to create the contract?

✔ Do you want to change the contract? [and so on]

✘ Do you want to `<variable 1>` `<variable 2>`? [Note: Concatenations with translatable words won't work in the target languages. Remember the thing with different word order and grammar?]

✔ Use method `<variable 1>` to call the document.

✘ Use method `<variable 1 (here: a technical name)>` `<variable 2>` `<variable 3>`. [Note: `<variable 2>` contains "to" and `<variable 3>` "call the document".]

✘ **German translation on the screen:** Verwenden Sie Methode Xyz zu Rufen Sie das Dokument auf. [Correct German: Verwenden Sie Methode Xyz, um das Dokument aufzurufen.]

✘ Use `<variable 1>` `<variable 2>`.

✘ Design Time with Concatenations	English at Run Time (Looks Fine But ...)	✘ Incorrect Czech at Run Time
✘ Use `<variable 1> <variable 2>`.	Use the preferred airline.	✘ Použijte preferován**o** letecká link**a**. [Note: Variables are translated without sentence context. Inflectional endings are not correct, that is, the sentence is not correct.] ✔ Použijte preferovan**ou** leteck**ou** link**u**. [Note: You'll only get this correct Czech sentence if it's translated from a complete English sentence, one without variables for translatable words.]
	Use the preferred hotel.	✘ Použijte preferován**o** hotel. ✔ Použijte preferovan**ý** hotel.
	Use the preferred vendor.	✘ Použijte preferován**o** dodavatel. ✔ Použijte preferovan**ého** dodavatel**e**.

Be careful with variables in number-noun combinations. Separate number and noun.

Constructions such as "`<variable>` days" do **not** work in other languages. Plural formation is not as regular as in English, numbers often require specific inflectional endings. For example, English has "1 day, 2 days, 5 days" versus Czech "1 den, 2 dny, 5 dní").

✤ Example

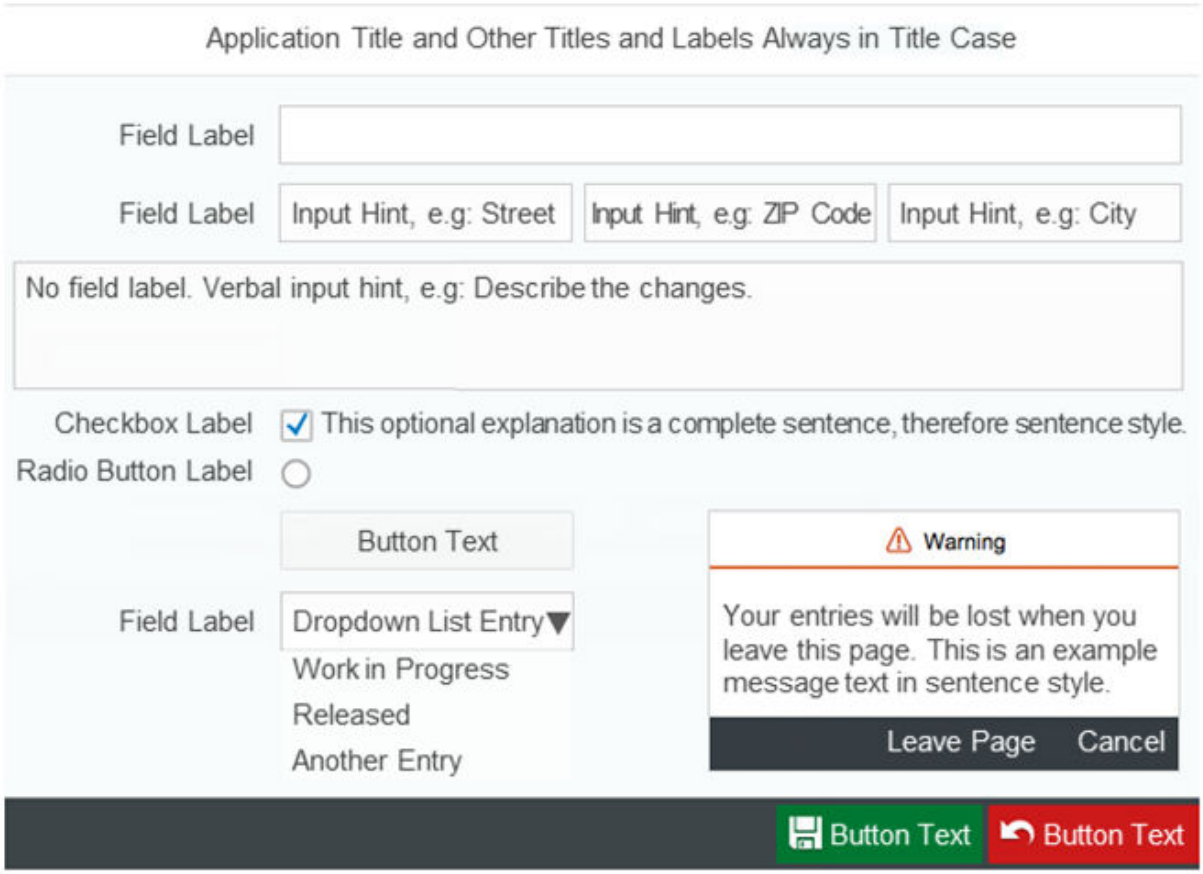
✔ Password expiration in days: `<variable (here: a number)>` [Note: This is the simplest solution to separate number and noun. But sometimes it might not fit in the overall text flow. Consult a localization expert from SAP Language Services.]

✘ This password expires in `<variable (here: a number)>` `<variable (here: hours, days, months)>`. [Note: Again, this is untranslatable, for reasons explained above.]

✘ Password expiration in day(s): `<variable (here: a number)>` [Note: "day(s)" – there won't be an equivalent in other languages. Plural formation is less regular. It's not just adding an "s" as in English.]

1.5 English: Capitalization (Sentence Style and Title Case) of UI Elements

Define guidelines for your applications and stick to these guidelines throughout your applications. It ensures consistency and adds a professional touch.



Use sentence style for messages and all other sentence-like text on the UI.

Sentence style is the normal, everyday capitalization style. It is the equivalent to using lowercase for all words, except for proper nouns and the first letter in a new sentence.

> ✤ Example
>
> ✔ Your booking will be reserved for 24 hours. [Note: This could be a confirmation message.]
>
> ✔ A draft of your last invoice was saved. Do you want to resume editing or discard the invoice? [Note: This is a user decision.]
>
> ✔ Classifies the installation for billing purposes [Note: This is a tooltip with sentence-like text. It briefly explains the *Rate Type* field. If your tooltip just gives the full form of an abbreviated field label such as *Rt. Tp.*, then the tooltip would be *Rate Type* in title case.]

Use title case for all types of titles and labels on the UI.

Title case applies to: field labels, checkboxes, radio buttons; buttons, menu items; predefined entries in a dropdown list; titles of screens, tabs, and dialog boxes; table titles and their column and row headings; and so on

How does title case work? Capitalize the **first letter** of the first and last word and of major words:

> ✤ Example
>
> ✔ Save File As, Approve Leave Request, My Documents, Purchase Order Overview, Frequently Bought Together, Work in Progress, Sold-to Party, Up-to-Date Information, About Us, Facts About the Product
>
> ✔ First and Last Words Get Title Case
>
> ✔ All Major Words (Nouns, Verbs, Adjectives, and Adverbs) Get Title Case
>
> ✔ I, My, You, Your, and Other Personal Pronouns Get Title Case
>
> ✔ This, That, These, Those, All, Any, Some, Every, Each, What, Where, Who, and Other Determiners Get Title Case
>
> ✔ Am, Is, Are, Was, Were, Be, Been, Has, Had, Do, Doing, Did, Create, Creating, Created, and Other Verbs Get Title Case
>
> ✔ About, After, Before, Between, During, Through, Under, Within, Without, and Other Prepositions with 5 or More Letters Get Title Case

1.6 Context Information for Translators

Any text or utterance in speech can be taken out of context. Avoid the risk by providing context information during the development phase. It will make the lives of translators and other experts who review translations much easier.

Help translators by providing notes, the text type, or self-descriptive text keys.

ABAP: In ABAP systems, the text type is automatically assigned. So nothing to worry about there.

Translators don't always get to see the application screens in the translation tool. Their worklist contains only text strings. To find the correct translation, they rely on your context information.

> ✤ Example
>
> [Note: Without context information, the English source text is impossible to translate. But if the translator knows that, for example, “Order” is a button text, he or she knows how to translate this text.]
>
> **English:** Order
>
> **German translation?** Bestellen [verb if it's a button] Auftrag [noun if it's a title or a label]
>
> **English:** Delete document

German translation? ?Dokument | Beleg? löschen [verb if it's a button] Löschen Sie ?das Dokument | den Beleg?. [complete sentence if it's a message]

Non-ABAP: Embed the context information in a comment.

You can use comments to set text types, text length restrictions (if absolutely necessary), and a note intended for the translator. Place the comment immediately before the key-value pair.

Use the following **pattern**: `# text type: note for translator`

Example

```
# XBUT: button text on the final Shopping Cart screen
view.SCADetail.Order=Place Order
```

```
# XTOL: welcome appears in the upper right corner of the screen, is followed
by first and last name
view.Shop.welcome=Welcome {0}
```

```
# XFLD: This is the start date.
view.LeaveRequest.StartDate=Start
```

[Note: If you **cannot avoid restricting the text length**, use the pattern `# text type, length restriction: note for translator` and provide a buffer for the other logon languages by applying the text length recommendations.]

```
# XFLD, 20: This is the start date.
view.LeaveRequest.StartDate=Start
```

Non-ABAP: Classify each UI text with the correct text type in the properties file.

The following table contains text types that SAP recommends:

Text Type	When to Use and Comments	Capitalization
`XMSG`	Message text Obsolete: `YMSG` (message text) and `XLOG` (log entry) → use `XMSG`	Sentence style applies: ✔ This is a text in sentence style.
`YINS`	Instruction for a user, for example, a permanent text on a screen that introduces a group of fields	
`XTOL`	Explanatory text for a UI element, for example, a tooltip	
`XLNK`	Hyperlink	
`XACT`	Accessibility text Text in the `alt` attribute is translated.	

Text Type	When to Use and Comments	Capitalization
`XTIT`	Title (or heading) of a nonactionable UI element, for example, a column heading, wizard title, or screen area title Obsolete: `XCOL` (column heading), `XGRP` (group title), and `XTBS` (tab title) → use `XTIT`	Title case applies (see also the section *English: Capitalization*): ✔ Titles and Labels in Title Case ✔ Rate Type ✔ Clean All Views ✔ Import Localized Content
`XFLD`	Field label	
`XBUT`	Button	
`XMIT`	Menu header or menu item	
`XSEL`	Values in a dropdown list Obsolete: `XLST` (list item) → use `XSEL`	Title case and sentence style as appropriate for text length: • Title case for short, label-like text, such as a status ✔ In Process • Sentence style for longer, more sentence-like text ✔ Workspaces closed in the last six months ✔ Check spelling as you type
`XCKL`	Checkbox label	
`XRBL`	Radio button label	

1.7 Wording and Formulations

Put yourself in the user's shoes when you write texts. Applications are much easier to use when UI texts focus on the task that users are currently processing.

Write from the user's point of view.

Use the active voice and “you”. Use positive formulations (“do”, rather than “don't”). Give clear instructions instead of describing the problem. Beware of long noun strings. Split a series of nouns into smaller, more manageable units. This is easier to understand and to translate.

> ✤ Example
>
> ✔ Enter the company code.
>
> ✘ Company code was not entered. [or:] BUKRS field is empty.
>
> ✔ You can create a worklist. [or:] Create a worklist.
>
> ✘ It is possible to create a worklist. [or:] A worklist can be created.
>
> ✔ After downloading the software, follow the instructions in the installation wizard.

❌ After the download an installer is available.

✔ Check the port signals from the device adapter card.

❌ Check the device adapter card port signals.

Do not use “(s)” or “/s” plurals.

Such constructions are hard to translate and will lead to awkward or unacceptable formulations in the target languages. Noun plurals are not always as simple and regular as in English. Some languages have no plural form. Other languages require multiple plural forms depending on the number preceding the noun. In many languages, noun plurals require verb and adjective agreement. Translators have to resort to a messy mix of parentheses and slashes.

Example

❌ **English:** Change the new contract(s).

❌ **German:** Ändern Sie den/die neuen Vertrag/Verträge.

❌ **French:** Modifiez le/les nouveau(x) contrat(s).

Here's how you can avoid these awkward formulations:

Example

✔ Select the regions for the sales revenue report. [Note: Often the plural form is sufficient.]

❌ Select the region(s) for the sales revenue report.

✔ Drag each file to the relevant folder.

❌ Drag the file(s) to the relevant folder(s).

✔ Number of documents deleted: `<number>` [Note: This works in all languages.]

❌ `<Number>` documents deleted [Note: This doesn't work in some target languages because they require multiple plural forms depending on the number preceding the noun.]

1.8 Typography and Punctuation Marks

Creating clear and concise texts make it easier to understand and use a UI and ensure that translation tools and translators can translate texts accurately without having to ask many questions.

Use a font size of 10 or higher.

In many Asian languages, characters with a font size smaller than 9 is difficult to read.

Use CamelCase or ALL CAPS only for technical names.

CamelCase and ALL CAPS are not easy to read. They also indicate to translators that this text must not be translated.

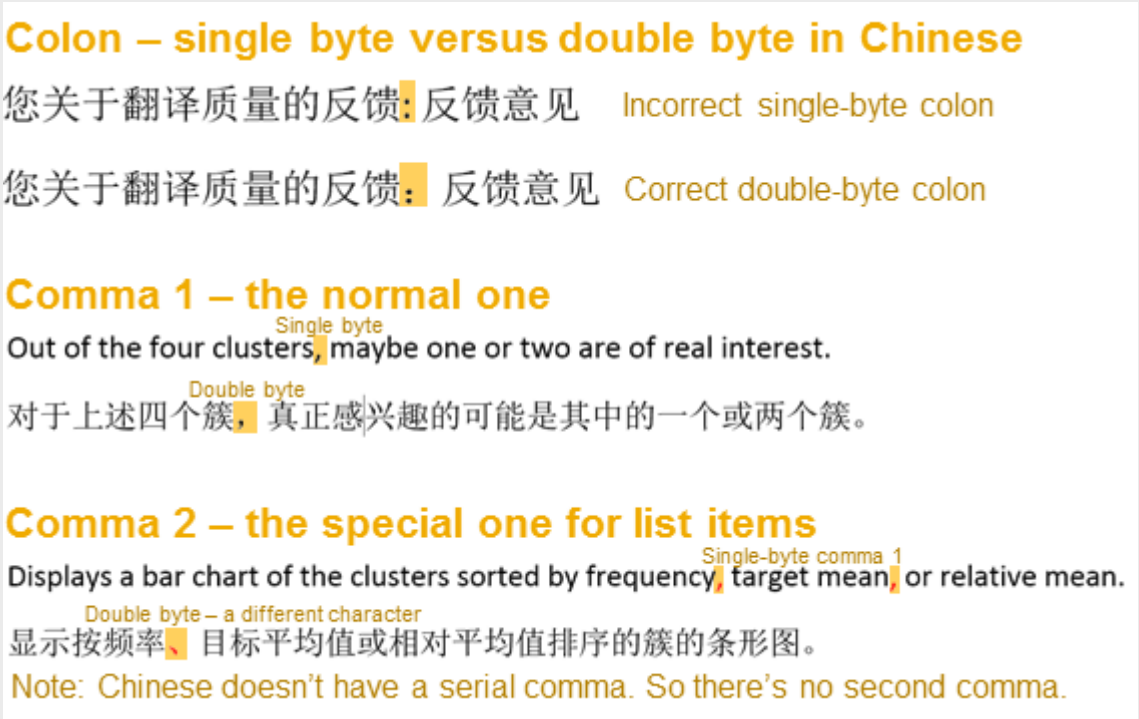
Make sure that punctuation marks, spaces, and special characters are part of a translatable text.

Do not hard code them. Do not add them at runtime. This enables translators to adapt the complete text. If you need help, consult a localization expert.

Why is this important? Punctuation rules, punctuation marks, and special characters (symbols) can be **surprisingly language-specific** with regard to use, typography, and word order. This is especially true for the **double-byte languages**, such as Chinese or Japanese. They do not use spaces as separators. Single-byte punctuation marks and special characters (symbols) have double-byte equivalents, which are typographically different. Sometimes there are even two double-byte equivalents, such as the two different Chinese commas. Sometimes there is no equivalent at all. For example, the number sign (#) is a widely used convention in the United States and Canada but not necessarily in other countries. Translators need to resort to a text equivalent such as “Number”, which will result in space issues on the user interface.

✤ Example

The correct Chinese double-byte colon is typographically different from the single-byte colon used in Western languages. The Chinese comma is actually two different commas, a normal one and a special one that's used as a separator in a list of items. Chinese speakers are very conscious of the difference. Incorrect use can lead to nonacceptance. Have a look at the following figure:



Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2025 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP