# A blue and black logo Description automatically generated

# Guidelines to translations in Strype

The process for translating Strype is simple. It relies on text files, and the reference for all translations is the English version.

This guide explains:

* [How to retrieve or create the translation files,](#_How_to_retrieve)
* [How to understand the translations files,](#_How_to_understand)
* [How to do the translations.](#_How_to_do)
* [How to visualise the translations on a test project.](#_How_to_visualise)
* [How to submit the translations.](#_How_to_submit)

*Note that currently, all translations are taking effect in Strype when a new version of the application is released by the Strype team. So even if a translation has been received by the Strype team and included into the source code repository, it will be visible in the online tool only when the Strype team releases a new version.*

## How to retrieve or create the translation files

To retrieve the translation files, you will need to locate them inside the Strype source code repository hosted in [GitHub](https://github.com/k-pet-group/Strype).   
The translation files for all supported locales are inside the /src/localisation/ folder, which contains **one subfolder per locale (as a 2-letter code)**. For example, the translations for English are in /src/localisation/**en**/, those of Spanish are in /src/localisation/**es**/ and so on.

A locale subfolder contains several translation JSON files, their names start with the 2-letter code of the locale (for example, en\_main.json). We split the translations in several files to ease the translation task and gather some specific translations separately (for example for Strype micro:bit).

You can either download a raw file from GitHub (by clicking on the desired file and clicking on the “download” button) OR you can use a web development IDE to work on the whole Strype project directly (cloning the project from GitHub with GIT). This option requires more preparation work (see the project’s [README](https://github.com/k-pet-group/Strype/) file for more details) **but will allow you to test your translation locally**.

If the locale you wish to provide translation for does not exist, you will need to create a new subfolder (name after the 2-letter code of the locale) and add your translation files following the same naming logic than mentioned above. The easiest way is to copy one of the en\_\*.json, rename it with the right locale code, and edit it with your translations.

Note: it is advised, if you want to retrieve a copy of the Strype source code, to first create a fork of the project into your own GitHub account and use your forked project as a remote GIT repository.

## How to understand the translations files

The translation files are [JSON](https://en.wikipedia.org/wiki/JSON) files: formatted text files that define ***key/value* pairs**. There are several files per language, identified by the corresponding locale and gathered inside one subfolder for that locale (see section [1](#_How_to_retrieve) for more details).

The *key* is used by Strype as *a reference* to a localisable text, independent of the locale.  
The corresponding *value* is the localised text *that will be displayed* in Strype, in a given locale.  
The *key* and its corresponding *value* are separated by a colon :.

Here is an example of a *key/value* pair for English, where the *key* (i.e. reference) used by Strype is **strypeFileDesc** and the *value* displayed in Strype is **Strype project file**:



In order to keep the *key/value* pairs logically organised, the locale files allow **nested definitions**: for example, all translations for a menu in Strype could appear as such:



In this case, the *value* of **appMenu** **is** **not a text**, but a list of nested other *key/value* pairs.   
(Nested definitions can comprise more than one level.)

**Only the text *values* need to be translated.** To differentiate a value that is a text from a value that represents a list of nested definitions, look at what follows the *key/value* separator colon: if that’s an opening curly bracket {, then the *value* is a list of nested definitions.

## How to do the translations

The language reference for all translations is the English locale. Therefore, the reference files are the files located in the /en/ subfolder (see sections [1](#_How_to_retrieve) and [2](#_How_to_understand) for details about finding the files and understanding their formats).

**Each** *key/value* pair of a translation locale file **need to match those of the equivalent English locale file**, and as explained above, **only** the text *values* need to be translated.

The easiest way to make sure that a translation locale file matches the equivalent English locale file is to open both files with a text editor, and perform a file comparison: ultimately, only the text *values* of both files should differ. We suggest [Notepad++](https://notepad-plus-plus.org/downloads/) for this, which contains a compare plugin.

Here are a few rules to follow when doing the translations of the text *values*.

* Leading and trailing spaces showing in the English text *values* **should be maintained** in the translated locale.
* Some text *values* contain Unicode characters starting with \u, followed by 4 characters. The *key* for these pairs is in this format: <Python keyword>\_detail. The Unicode characters represent a Python keyword.
  + The Unicode characters **should not be altered** in the translation.
  + A translation of the Python keyword can be added into brackets for the given locale.

For example, the English reference contains this *key/value* pair:  


that can be translated in another locale as such:  
The Unicode characters stay the same in both locales, and **(si)** has been added in the translation to provide the literal translation of the Python keyword too.

* Some text *values* contain HTML tags or codes (such as <b> or &larr;). They are interpreted as such by Strype to style the text in the browser, and therefore **should** **stay in the translation**. However, the text content between the opening and closing tags can be translated if that makes sense.
* Some text *values* contain curly brackets surrounding a word. This are used by Strype as placeholders to dynamically add pieces of text within the translation, and therefore the content between the curly brackets **should not be translated**.
* Some text *values* contain an escaped character starting with \ which **should** **stay in the translation**. For example, the English reference contains this *key/value* pair:  
  

That can be translated in another locale as such:  


The escape character \n is left in the translation.

* For the micro:bit documentation translation, the English reference is based on the official micro:bit documentation (<https://microbit-micropython.readthedocs.io>) which may be available in other languages.

In case of doubts, please contact us ([team@strype.org](mailto:team@strype.org)) for explanations.

## How to visualise the translations on a test project

The only way to see how your translations are rendered in Strype is to download the project source code and run it. See section [1](#_How_to_retrieve) for details on where to retrieve the Strype source code. fIf you have edited your translations *outside* the code source (for example if started working on a translation before downloading and installing the Strype project), you need to make sure that your changes are applied into your local copy of Strype before running the project (check the project’s [README](https://github.com/k-pet-group/Strype/) for the procedure).

*Tip: when you want to check which translations are missing from Strype, run the project as explained above, and once your local version of the Strype application is opened with your Internet browser, open the development tool (how?). You will see each SON keys that are missing a value. Example:*

*A screenshot of a computer

Description automatically generated*

## How to submit the translations

There are 2 ways you can submit your translations: either sending the locale file(s) to us directly ([team@strype.org](mailto:team@strype.org)) or, if you have downloaded and installed a local copy of the Strype source code (see section [1](#_How_to_retrieve), you can commit your changes (with GIT), push your changes to the remote GIT repository and create a Pull Request on GitHub for us to review your changes.

NOTE: since English is the reference language used by Strype, *any missing translations in a locale will be rendered in Strype in English*. This means that if you prefer sending us your translations in several steps (for example, a file at a time), Strype will still be able to run with some missing translations.