

# Harmattan - Developer information

## Welcome font developers!

We welcome other developers who wish to get involved in supporting and enhancing these fonts or who want to modify them.

## Permissions granted by the OFL

SIL's fonts are licensed according to the terms of the [SIL Open Font License](#). The OFL allows the fonts to be used, studied, modified and redistributed freely as long as they are not sold by themselves. For details see the OFL.txt and OFL-FAQ.txt files in the package.

## Building the fonts from source code

Font sources are published in a [Github project](#). The build process requires [smith](#) and project build parameters are set in the [wscript](#).

Font sources are in the [UFO3](#) format with font family structures defined using [designspace](#). OpenType source code is stored in the [.fea](#) format in the UFO (features.fea) but is maintained in a separate file using the more efficient and powerful [.feax](#) format.

The fonts are built using a completely free and open source workflow using industry-standard tools ([fonttools](#)), a package of custom python scripts ([pysilfont](#)), and a build and packaging system ([Smith](#)). The whole toolchain is available as a Docker container.

Full instructions for setting up the tools and building SIL fonts are available on a dedicated web site: [SIL Font Development Guide](#). Additional developer information specific to SIL's Arabic fonts can be found at [font-arab-tools README](#).

In addition, much of the code for Scheherazade New, Harmattan, and Lateef is shared. Carefully review the [font-arab-tools developer](#) documentation to see how the code is shared.

## Building

The Harmattan project can be built from source using [smith](#). This is done via the sequence:

```
smith distclean
smith configure
smith build
smith alltests
```

Because of the complex kerning and collision avoidance logic, builds can take up to 15 minutes or longer, depending on hardware. If the complex kerning is *not* needed (such as for debugging other font logic), the `--quick` parameter can be supplied:

```
smith distclean
smith configure
smith build --quick
```

The resulting files will not have functional kerning or collision avoidance, but will be otherwise usable.

This project implements two additional `smith build` options that are useful during development:

- `--regOnly` -- build only the Regular weight instead of all weights
- `--norename` -- keep the working names for glyphs rather than change them to production names

Finally, to include Graphite smarts (which are used during development for generating kerning rules -- see `updateKerning.sh` below), use the `--graphite` option as follows:

```
smith distclean
smith configure --graphite
smith build --graphite --quick --norename
```

## Adding characters

After adding glyphs (other than used only as components for building other glyphs) to the font, the following files will also need updating:

- `glyph_data.csv` -- used to set glyph orders and psnames in the built font
- `classes.xml` -- used to define classes used by both OpenType and Graphite. Note that some of the classes defined therein are marked "automatically generated" -- these will be updated (from `glyph_data.csv`) the next time `./preflight` is run.
- `opentype/*.feax` -- modify as needed to add needed OpenType behavior
- `graphite/*.gd*` -- modify as needed to add needed Graphite behavior
- `tests/*.ftml` -- see below

Additionally the `*-octabox.json` files will need to be regenerated in order to add optimal octaboxes for the isolate and initial forms of the newly added characters. If unencoded variants of isolate and initial forms have been added, these must be manually added to the `cComplexShape` class defined in `source/graphite/caBasedKerning.gdl` so they get optimized as well.

## Generated source files

Five of the source files needed for the build are actually generated files but, because they require compute-intensive tools to create or update, are generated offline and committed to the repo. The files that fall into this category are:

- `source/kerndata.ftml` -- contains strings with all possible combinations of reh-like and following initials or isolates. This is used to extract graphite collision-avoidance-based kerning data.
- `source/graphite/*-octabox.json` -- optimized octaboxes to enable Graphite to do more accurate kerning of reh-like characters to what follows.
- `source/opentype/caKern-*.fea` -- contextual kerning rules that approximate the kerning effected by the Graphite collision avoidance.

If the *design* of any Arabic glyphs in the font changes, it is important to:

- rebuild the optimized `octabox.json` files so that Graphite collision-avoidance-based kerning is accurate, and then
- rebuild the OpenType kerning rules from the graphite results.

A script to do this is in `tools/updateKerning.sh`. This should be run from the root of the project. Be aware this can take up to 30 minutes or more to complete.

Important notes: The `updateKerning.sh` tool requires:

- fully functioning `smith` build system
- a Graphite-enabled Harfbuzz library
- a Graphite library with tracing enabled. The library provided by default Ubuntu does not include tracing. You'll need to compile the source with `-DGRAPHITE2_NTRACING:BOOL=OFF`
- the `scikit-learn` python module. For ubuntu try:

```
sudo apt-get install python3-sklearn python3-sklearn-lib
```

To generate `kerndata.ftml` and the `*-octabox.json` files, run:

```
tools/updateKerning.sh --ftml --octalap
```

## Generated test files

After adding characters or additional behaviors to the font, test files should be created or enhanced to test the new behaviors. The test files:

- tests/AllChars-auto.ftml
- tests/ALsorted-auto.ftml
- tests/DaggerAlef-auto.ftml
- tests/DiacTest1-auto.ftml
- tests/DiacTest1-short-auto.ftml
- tests/FeatLang-auto.ftml
- tests/Kern-auto.ftml
- tests/SubtendingMarks-auto.ftml
- tests/Yehbarree-auto.ftml

are generated automatically using `tools/genftmlfiles.sh`. This script, in turn, calls `tools/absgenftml.py` to create each test file. A lot of test generation logic is driven by Unicode character properties and the `glyph_data.csv` file, but sometimes `absgenftml.py` itself needs to be enhanced.

For more information about testing, see [font-arab-tools testing](#).

## Contributing to the project

We warmly welcome contributions to the fonts, such as new glyphs, enhanced smart font code, or bug fixes. The [brief overview of contributing changes](#) is a good place to begin. The next step is to contact us by responding to an existing issue or creating an issue in the Github repository and expressing your interest. We can then work together to plan and integrate your contributions.

To enable us to accept contributions in a way that honors your contribution and respects your copyright while preserving long-term flexibility for open source licensing, you would also need to agree to the **SIL Global Contributor License Agreement for Font Software (v1.0)** prior to sending us your contribution. To read more about this requirement and find out how to submit the required form, please visit the [CLA information page](#).

This guide is from the [Harmattan project](#) version 4.4 and is copyright © 2014-2025 SIL Global.