

Garmin POI Builder

Downloads pass images and GPX data from passknacker.com, processes them, and creates Garmin GPI POI files — one per country.

Requirements

- Python 3.10 or higher
- [Pillow](#)
- [requests](#)

Install dependencies:

```
pip install -r requirements.txt
```

Or install the package directly (also installs dependencies):

```
pip install -e .
```

Setup

1. Copy `config.json` to your working directory and adjust as needed (see [Configuration](#))
-

Usage

Without installation (run from project root)

```
python -m garmin_poi_builder run
python -m garmin_poi_builder run --from 3 --to 7
python -m garmin_poi_builder correct-gpx
```

With installation (`pip install -e .`)

```
gpb run
gpb run --from 3 --to 7
gpb correct-gpx --config /path/to/config.json
```

All commands accept `--config FILE` to point to a different config file (default: `config.json` in the current directory).

Pipeline Steps

The pipeline consists of 7 steps that can be run all at once or individually.

Full pipeline

```
gpb run
```

Runs all steps 1–7 in sequence. Each step prompts for required input before executing.

Partial pipeline

```
gpb run --from 3 --to 5
```

Runs steps 3, 4 and 5 only. Useful when resuming after an interruption or when only part of the pipeline needs to be repeated.

Individual steps

Command	Step	Description
<code>gpb login</code>	1	Login to passknacker.com
<code>gpb download-images</code>	2	Download country image ZIPs and extract
<code>gpb download-gpx</code>	3	Download master GPX file
<code>gpb check-images</code>	4	Check GPX image links, download missing
<code>gpb compress-images</code>	5	Compress and resize images
<code>gpb correct-gpx</code>	6	Correct GPX (sym, proximity, hrefs)
<code>gpb create-gpi</code>	7	Create Garmin GPI file(s)

Step Details

Step 1 – Login

Prompts for your passknacker.com nickname and password (password input is hidden). The authenticated session is passed to all subsequent steps.

Step 2 – Download images

Prompts for an output directory. Downloads one ZIP archive per country and extracts images into `<output_dir>/<images_original>/`. ZIPs are saved directly in `<output_dir>/`.

Supports **dry-run** mode: existing ZIPs are skipped.

Step 3 – Download GPX

Prompts for export options (countries, selection, driven status). Downloads the master GPX file into the same output directory as the ZIPs. The filename is taken from the server's Content-Disposition header.

Supports **dry-run** mode: skips download if the file already exists.

Step 4 – Check images

Parses the GPX file and checks every `<link href="...">` against the local images folder. Downloads any missing images from passknacker.com.

Supports **dry-run** mode: existing files are reported but not re-downloaded.

Step 5 – Compress images

Resizes and compresses all images from the original images folder into a new `images_compressed/` folder. Size and quality come from `config.json`.

Supports **dry-run** mode: existing files in the output folder are skipped.

Step 6 – Correct GPX

Applies the following corrections to the GPX file:

- Sets `<sym>ATV</sym>` for waypoints with category LK: XXX
- Sets `<Proximity>` on all waypoints to the prompted distance
- Rewrites all `<link href="...">` to point to the compressed images folder
- Normalises backslashes to forward slashes (Linux/macOS)

Writes a new `_corrected.gpx` file, leaving the original untouched.

Step 7 – Create GPI

Converts the corrected GPX + compressed images into Garmin GPI files — one per country code found in the GPX, plus an optional combined ALL file.

Prompts for filename stem, version suffix, output directory and whether to write a combined ALL.gpi.

Output Folder Structure

```
<output_dir>/
├─ CHE_passbilder.zip           ← step 2: downloaded ZIPs
├─ ITA_passbilder.zip
├─ ...
├─ ALL_Passknacker_2026.gpx    ← step 3: downloaded GPX
├─ ALL_Passknacker_2026_corrected.gpx ← step 6: corrected GP
├─
├─ images_original/           ← step 2: extracted original in
│   ├── pass_001.jpg
│   ├── pass_002.jpg
│   └─ ...
├─
├─ images_compressed/         ← step 5: resized/compressed in
│   ├── pass_001.jpg
│   ├── pass_002.jpg
│   └─ ...
├─
└─ gpi/                       ← step 7: Garmin GPI output
    ├── CHE_PASSKNACKER_2026_V1.gpi
    ├── ITA_PASSKNACKER_2026_V1.gpi
    └─ ...
```

Configuration

All settings live in `config.json`. The `{base_url}` placeholder in URL values is resolved automatically at startup.

```

{
  "base_url": "https://www.passknacker.com",
  "login_url": "{base_url}/login.php",
  "pictures_url": "{base_url}/pictures/pass/packs/{_passbilde
  "picture_url": "{base_url}/pictures/pass/",
  "gpx_url": "{base_url}/export_master_sel.php?pic=1",

  "gpx_payload_defaults": {
    "laender": "ALL",
    "auswahl": "all",
    "gefahren": "both",
    "filetype": "GPX",
    "geographisch": ""
  },

  "gpx_proximity": 1000.0,
  "images_original": "images_original",
  "image_compress_size": [500, 375],
  "image_compress_quality": "low",

  "gpi_max_icon": 22,
  "gpi_trans_color": [254, 254, 254],
  "gpi_format": 0,
  "gpi_source": "gpx2gpi",

  "countries": ["CHE", "ITA", "FRA", "AUT", ...]
}

```

Key	Description
base_url	Base URL of passknacker.com
pictures_url	URL template for country ZIP downloads ({_} = country code)
picture_url	Base URL for individual image downloads
gpx_url	GPX export endpoint
gpx_payload_defaults	Default POST parameters for GPX export
gpx_proximity	Default proximity alert distance in metres (step 6)
images_original	Subfolder name for extracted original images (step 2)
image_compress_size	[width, height] max thumbnail size in pixels (step 5)

image_compress_quality	JPEG quality preset: low (40), mid (60), high (80) (step 5)
gpi_max_icon	Max icon side in pixels for GPI bitmaps (step 7)
gpi_trans_color	Transparent colour [R, G, B] for GPI icons (step 7)
gpi_format	GPI format version: 0 (cp1252) or 1 (UTF-8) (step 7)
gpi_source	Data-source label shown in Garmin device (step 7)
countries	List of ISO 3166-1 alpha-3 country codes to process

Package Structure

```
garmin_poi_builder/
├─ pyproject.toml
├─ requirements.txt
├─ config.json
├─ garmin_poi_builder/
│   ├─ __init__.py
│   ├─ __main__.py
│   ├─ main.py           ← pipeline runner
│   ├─ cli.py            ← argparse entry points
│   ├─ core/
│   │   └─ gpx2gpi.py    ← GPX → GPI conversion engine
│   └─ resources/
│       ├─ noIMG.jpg      ← fallback image (place here manu
│       ├─ flag_red_22.png
│       ├─ flag_red_32.png
│       ├─ flag_red_48.png
│       ├─ flag_green_22.png
│       ├─ flag_green_32.png
│       ├─ flag_green_48.png
│       ├─ dirtbike_22.png
│       ├─ dirtbike_32.png
│       └─ dirtbike_48.png
├─ steps/
│   ├─ login.py
│   ├─ download_images.py
│   ├─ download_gpx.py
│   ├─ check_images.py
│   ├─ compress_images.py
│   ├─ correct_gpx.py
│   └─ create_gpi.py
└─ utils/
    ├─ config.py
    └─ session.py
```

Notes

- Credentials are never stored — they are prompted once per run via `getpass`
- Dry-run mode is available in steps 2, 3, 4 and 5 to skip already-downloaded/processed files
- Steps share runtime state (paths, session) through the config dict during a full `gpb` run. When running steps individually, each step will prompt for any paths it needs

Credits

Written by Matthias Hirsch (RedXR) and Claude (Anthropic)

This tool is for personal use with passknacker.com. Use is subject to passknacker.com's terms of service.