

# GenPPI Deployment Guide for PyPI

GenPPI Team

June 24, 2025

This document is a detailed guide on how to prepare, test, and publish the GenPPI Python interface on the Python Package Index (PyPI). Version 0.1.3 uses the py7zr library for multi-volume file extraction to ensure compatibility.

## 1 Preparation Checklist

Before publishing, follow this checklist to ensure the package is ready.

### 1.1 Check Package Structure

Your project structure should look like this:

```
genppi_py/
|-- genppi_py/
|   |-- bin/
|   |   |-- __init__.py # Important to treat 'bin' as a submodule
|   |-- __init__.py
|   |-- cli.py
|   |-- genppi.py
|   |-- download_model.py
|   |-- download_model_direct.py
|   |-- download_samples.py
|-- setup.py
|-- pyproject.toml
|-- MANIFEST.in
|-- README.md
|-- LICENSE
|-- MAINTAINERS.md
|-- prepare_release.py
|-- test_genppi.py
|-- deployment.tex
|-- testing.tex
|-- install.bat
|-- install.sh
```

### 1.2 Update Package Version

Edit the `genppi_py/__init__.py` and `pyproject.toml` files to update the version number, following semantic versioning.

```
1 # in genppi_py/__init__.py
2 __version__ = '0.1.3' # Current version
3
```

```

4 # in pyproject.toml
5 [project]
6 name = "genppi-py"
7 version = "0.1.3"
8 # ... rest of the configuration

```

## 1.3 Review Configuration Files

### 1.3.1 setup.py

Version 0.1.3 uses a simplified configuration. Required dependencies are defined in `pyproject.toml`:

```

1 # Current setup.py configuration (simplified)
2 setup(
3     cmdclass={
4         'install': CustomInstall,
5     },
6 )
7
8 # Dependencies are defined in pyproject.toml
9 [project]
10 dependencies = [
11     "py7zr>=0.20.0,<1.0.0", # For multi-volume 7z extraction
12     "multivolume>=0.2.3", # Required by py7zr
13 ]

```

Version 0.1.3 ensures that critical dependencies (py7zr and multivolume) are installed automatically. This eliminates issues with multi-volume file extraction.

#### Important notes on licenses:

- Ensure the package name in `README.md` matches the name in `setup.py` (genppi-py).
- **Issue with modern license fields:** TestPyPI does not support fields like `license_files` or `license-expression`. To avoid upload errors, use only the license classifier in `classifiers`.
- If you encounter a "unrecognized or malformed field 'license-file'" error, temporarily remove the `LICENSE` file during the build, or just use the classifier.
- For maximum compatibility, only use: `'License :: OSI Approved :: GNU General Public License v3 (GPLv3)'` in the classifiers.

### 1.3.2 pyproject.toml

This file contains the modern project configuration, including metadata and dependencies:

```

1 [build-system]
2 requires = ["setuptools>=42", "wheel"]
3 build-backend = "setuptools.build_meta"
4
5 [project]
6 name = "genppi-py"
7 version = "0.1.3"
8 description = "Python interface for GenPPI"
9 dependencies = [
10     "py7zr>=0.20.0,<1.0.0",
11     "multivolume>=0.2.3",
12 ]
13

```

```

14 [project.scripts]
15 "genppi.py" = "genppi_py.genppi:main"
16 "genppi-download-samples" = "genppi_py.download_samples:
    download_samples"
17 "genppi-download-model" = "genppi_py.download_model:main"

```

## 1.4 Generate Distribution Files

Install the build tools and create the distribution packages.

```

1 # Install the tools if you haven't already
2 pip install build twine
3
4 # Clean old builds to avoid issues
5 rm -rf dist/ build/ *.egg-info
6
7 # Create the new distribution packages
8 python -m build

```

This command will create a `dist/` folder containing a `.tar.gz` file (source distribution) and a `.whl` file (wheel).

## 2 Publishing to PyPI

With the packages generated, the next step is to publish them.

### 2.1 Step 1: Publish to TestPyPI (Highly Recommended)

Always publish to the test server first to ensure everything works.

```

1 # Upload to TestPyPI
2 python -m twine upload --repository testpypi dist/*

```

You will need an account on [TestPyPI](https://test.pypi.org/). After uploading, test the installation in a clean virtual environment:

```

1 # Create and activate a new virtual environment
2 python -m venv test_env
3 source test_env/bin/activate
4
5 # Install version 0.1.3 from TestPyPI (with dependencies from official
    PyPI)
6 pip install --index-url https://test.pypi.org/simple/ --extra-index-url
    https://pypi.org/simple/ genppi-py==0.1.3
7
8 # Verify critical dependencies
9 python -c "import py7zr; print('py7zr:', py7zr.__version__)"
10 python -c "import multivolumefile; print('multivolumefile OK')"
11
12 # Run a quick test
13 genppi.py --help

```

### 2.2 Step 2: Publish to the Official PyPI

**IMPORTANT - TestPyPI vs. Official PyPI:**  
**TestPyPI (For Developers Only):**

- A test environment with limited dependencies.
- Requires a complex command to install dependencies.
- End-users should NOT use this command.

#### Official PyPI (For End-Users):

- A production environment with all dependencies.
- Uses a simple command for installation.
- Provides an optimized user experience.

If the TestPyPI trial was successful, publish to the official PyPI:

```
1 # Upload to the official PyPI
2 python -m twine upload dist/*
3
4 # Or use an automated script
5 python deploy_production.py
```

**Result for the End-User:** After the official release, users can install simply with:

```
1 # SIMPLE COMMAND FOR END-USERS
2 pip install genppi-py
```

You will need an account on [PyPI](#).

## 2.3 Step 3: Final Verification

After publishing, check your project page at <https://pypi.org/project/genppi-py/>. Finally, perform one last installation test from the official PyPI in a clean virtual environment to ensure a smooth end-user experience.

## 2.4 Common Troubleshooting

### 2.4.1 Error: unrecognized or malformed field 'license-file'

This error occurs when setuptools generates modern license fields that PyPI does not yet support. To resolve it:

1. Temporarily remove the LICENSE file from the project directory.
2. Use only the traditional license classifier in `setup.py`.
3. Avoid using `license_files` or `license` fields in `pyproject.toml`.
4. After a successful upload, restore the LICENSE file.

### 2.4.2 Verifying Metadata

Before uploading, always check the generated metadata:

```
1 # Check the metadata content
2 cat genppi_py.egg-info/PKG-INFO | head -20
3
4 # Look for problematic fields such as:
5 # License-File: LICENSE
6 # License-Expression: GPL-3.0-or-later
```

### 2.4.3 Dependency Issues on TestPyPI

Version 0.1.3 uses dependencies available on the official PyPI. If you encounter issues:

```
1 # Check if py7zr is available
2 pip install "py7zr>=0.20.0,<1.0.0"
3 pip install "multivolumefile>=0.2.3"
4
5 # Then install the test package (with dependencies from official PyPI)
6 pip install --index-url https://test.pypi.org/simple/ --extra-index-url
   https://pypi.org/simple/ genppi-py==0.1.3
7
8 # Verify installation
9 python -c "import py7zr, multivolumefile; print('Dependencies OK')"
```

## 3 Automation and Maintenance

### 3.1 Automated Deployment Scripts

The project includes automated scripts to streamline the deployment process:

#### 3.1.1 quick\_check.py

Performs a quick check of dependencies and features:

```
1 python quick_check.py
```

#### 3.1.2 build\_and\_test.py

Runs a full build with comprehensive tests:

```
1 python build_and_test.py
```

#### 3.1.3 deploy\_test.py

Automates deployment to the test environment:

```
1 python deploy_test.py
```

### 3.2 Manual Publishing Script

For manual releases, you can use a script like `publish.sh`:

```
1 #!/bin/bash
2 set -e # Exit script if a command fails
3
4 # Ask for version confirmation
5 read -p "Have you updated the version in __init__.py? (y/n) " -n 1 -r
6 echo
7 if [[ ! $REPLY =~ ^[Yy]$ ]]; then
8     exit 1
9 fi
10
11 echo "Cleaning old builds..."
12 rm -rf dist/ build/ *.egg-info
13
```

```

14 echo "Generating new packages..."
15 python -m build
16
17 echo "Uploading to PyPI..."
18 python -m twine upload dist/*
19
20 echo "Publication complete!"

```

### 3.3 Updating the Package

To release a new version:

1. Update the source code with new features or fixes.
2. If needed, update the Lisp executables or `model.dat` file in the source GitHub repository.
3. Increment the version number in `genppi_py/__init__.py`.
4. Run the `publish.sh` script to publish the new version.

### 3.4 Updating `model.dat`

If you need to generate new versions of `model.dat`:

1. Create the new `model.dat` file.
2. Compress it into parts using 7-Zip (e.g., 10MB parts):

```

1 7z a -v10m model.7z model.dat
2

```

3. Test the extraction with `py7zr`:

```

1 python -c "
2 import py7zr
3 from multivolumefile import MultiVolume
4 with MultiVolume('model.7z', mode='rb') as vol:
5     with py7zr.SevenZipFile(vol, mode='r') as archive:
6         print('Files in volume:', archive.getnames())
7         archive.extractall(path='test_extract')
8 "
9

```

4. Replace the `model.7z.00*` files in your source repository.