GenPPI Deployment Guide for PyPI

GenPPI Team

June 26, 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 0.1.7 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/
|-- dev.py
|-- docs
|   |-- deployment.pdf
|   |-- deployment.tex
|   |-- README.md
|   |-- testing.pdf
|   `-- testing.tex
|-- genppi_py
|   |-- bin
|   |   `-- __init__.py
|   |-- cli.py
|   |-- download_model_direct.py
|   |-- download_model.py
|   |-- download_samples.py
|   |-- genppi.py
|   `-- __init__.py
|-- install.bat
|-- install.sh
|-- MAINTAINERS.md
|-- MANIFEST.in
|-- pyproject.toml
|-- README.md
|-- scripts
|   |-- build_and_test.py
|   |-- deploy_production.py
|   |-- deploy_test.py
|   |-- quick_check.py
|   |-- README.md
|   |-- test_installation.py
|   `-- test_upload.py
|-- setup.py
|-- tests
|   |-- README.md
|   `-- test_genppi.py
`-- tools
    |-- prepare_release.py
    |-- README.md
    |-- requirements-dev.txt
    `-- tox.ini
```

1.2 Update Package Version

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

```
# in genppi_py/__init__.py
__version__ = '0.1.7'  # Current version

# in pyproject.toml
[project]
name = "genppi-py"
version = "0.1.7"
# ... rest of the configuration
```

1.3 Review Configuration Files

1.3.1 setup.py

Version 0.1.7 uses a simplified configuration with Python 3.8+ requirement. Required dependencies are defined in `pyproject.toml`:

```
# Current setup.py configuration (simplified)
setup(
    cmdclass={
        'install': CustomInstall,
    },
)

# Dependencies are defined in pyproject.toml
[project]
dependencies = [
    "py7zr>=0.20.0,<1.0.0",  # For multi-volume 7z extraction
    "multivolumefile>=0.2.3",  # Required by py7zr
]
```

Version 0.1.7 ensures that critical dependencies (py7zr and multivolumefile) are installed automatically with Python 3.8+ requirement. This eliminates issues with multi-volume file extraction and compatibility problems.

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:

```
[build-system]
requires = ["setuptools>=42", "wheel"]
build-backend = "setuptools.build_meta"

[project]
name = "genppi-py"
version = "0.1.7"
description = "Python interface for GenPPI"
dependencies = [
    "py7zr>=0.20.0,<1.0.0",
    "multivolumefile>=0.2.3",
]

[project.scripts]
"genppi" = "genppi_py.genppi:main"
"genppi-download-samples" = "genppi_py.download_samples:
    download_samples"
"genppi-download-model" = "genppi_py.download_model:main"
```

1.4 Generate Distribution Files

Install the build tools and create the distribution packages.

```
# Install the tools if you haven't already
pip install build twine

# Clean old builds to avoid issues
rm -rf dist/ build/ *.egg-info

# Create the new distribution packages
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.

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

You will need an account on TestPyPI. After uploading, test the installation in a clean virtual environment:

```
# Create and activate a new virtual environment
python -m venv test_env
source test_env/bin/activate

# Install version 0.1.7 from TestPyPI (with dependencies from official
   PyPI)
pip install --index-url https://test.pypi.org/simple/ --extra-index-url
    https://pypi.org/simple/ genppi-py==0.1.7

# Verify critical dependencies
python -c "import py7zr; print('py7zr:', py7zr.__version__)"
python -c "import multivolumefile; print('multivolumefile OK')"

# Run a quick test
genppi --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:

```
# Upload to the official PyPI
python -m twine upload dist/*

# Or use an automated script
python deploy_production.py
```

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

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

# After installation, use the 'genppi' command (not 'genppi.py')
genppi --help
genppi -dir samples
```

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:

```
# Check the metadata content
cat genppi_py.egg-info/PKG-INFO | head -20

# Look for problematic fields such as:
# License-File: LICENSE
# License-Expression: GPL-3.0-or-later
```

2.4.3 Dependency Issues on TestPyPI

Version 0.1.7 uses dependencies available on the official PyPI. If you encounter issues:

```
# Check if py7zr is available
pip install "py7zr>=0.20.0,<1.0.0"
pip install "multivolumefile>=0.2.3"

# Then install the test package (with dependencies from official PyPI)
pip install --index-url https://test.pypi.org/simple/ --extra-index-url
    https://pypi.org/simple/ genppi-py==0.1.7

# Verify installation
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:

```
python quick_check.py
```

3.1.2 build_and_test.py

Runs a full build with comprehensive tests:

```
python build_and_test.py
```

3.1.3 deploy_test.py

Automates deployment to the test environment:

```
python deploy_test.py
```

3.2 Manual Publishing Script

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

```bash
#!/bin/bash
set -e # Exit script if a command fails

# Ask for version confirmation
read -p "Have you updated the version in __init__.py? (y/n) " -n 1 -r
echo
if [[ ! $REPLY =~ ^[Yy]$ ]]; then
    exit 1
fi

echo "Cleaning old builds..."
rm -rf dist/ build/ *.egg-info

echo "Generating new packages..."
python -m build

echo "Uploading to PyPI..."
python -m twine upload dist/*

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 `pyproject.toml`.
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):

   ```
   7z a -v10m model.7z model.dat

   ```

3. Test the extraction with py7zr:

```
python -c "
import py7zr
from multivolumefile import MultiVolume
with MultiVolume('model.7z', mode='rb') as vol:
    with py7zr.SevenZipFile(vol, mode='r') as archive:
        print('Files in volume:', archive.getnames())
        archive.extractall(path='test_extract')
"

```

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