

Data Exchange Workflow for **dvi3v** project

v3.9.2.208

Peter Felecan

with Claude (Anthropic AI), model Sonnet 5

October 12, 2026

Contents

1	Purpose	1
2	The Two Archive Types	2
2.1	Project Source Archive (<code>CreateProjectArchive</code>)	2
2.2	Test Data Archive (<code>CreateTestDataArchive</code>)	2
3	Naming Convention	2
4	Typical Workflow	3
4.1	Producing Test Data	3
4.2	Using Test Data	3

List of Figures

List of Tables

1 Purpose

The project’s source (the tool, its documentation, its build system) and its test execution data (photographs, generated SVGs, debug images) are packaged and shared **separately**, as two different archive types. Test execution data can be large and grows with every test run; bundling it into every source release would make ordinary version updates unwieldy, and most consumers of a release do not need it. Two purpose-built scripts, described below, produce each archive type.

2 The Two Archive Types

2.1 Project Source Archive (**CreateProjectArchive**)

Bundles the project’s source: the tool, its `.in` templates, the Org-mode documentation set, the build system, hardware/test-object design files. Run from the project root with no arguments:

```
./CreateProjectArchive
```

Produces `dvi3v-VERSION.tar.xz` (the version read from the project’s own `VERSION` file), excluding generated build artefacts (PDFs, HTML, `version.org`, $\text{\LaTeX}$ auxiliary files, `__pycache__`) and test execution data, per `.tar-exclude`. This is the archive delivered with each project version.

2.2 Test Data Archive (**CreateTestDataArchive**)

Bundles `TestResults/` — the photographs, generated SVGs, and debug images from actual test runs. Run from the project root:

```
# Everything under TestResults/
./CreateTestDataArchive
```

```
# Only one category, e.g. TestResults/RealWorld/SeamRipper
./CreateTestDataArchive -c RealWorld/SeamRipper
```

Produces a timestamped `dvi3v-TestData-[CATEGORY-]YYYY-MM-DD-HHMM.tar.xz`, together with `TestResults/VERSION.org` (created from a template on first run if it does not exist, recording which project version the test data belongs to — edit it before sharing the archive). See `man CreateTestDataArchive` for the full option reference.

This is the archive to produce whenever test results, or a specific test case, need to be shared or reviewed — including when reporting a problem; see [Debug File Upload](#) for what to capture in that case specifically.

3 Naming Convention

Archive type	Pattern	Example
Project source	<code>dvi3v-VERSION.tar.xz</code>	<code>dvi3v-3.9.1.tar.xz</code>
Test data (all)	<code>dvi3v-TestData-YYYY-MM-DD-HHMM.tar.xz</code>	<code>dvi3v-TestData-2026-08-12</code>
Test data (category)	<code>dvi3v-TestData-CATEGORY-YYYY-MM-DD-HHMM.tar.xz</code>	<code>dvi3v-TestData-RealWorld_S</code>

A `/` in a given `-c CATEGORY` becomes `_` in the filename (a category path such as `RealWorld/SeamRipper` becomes `RealWorld_SeamRipper` in the archive name), since the archive is a single flat file.

4 Typical Workflow

4.1 Producing Test Data

1. Run `dvi3v` with `--debug` on the case(s) of interest, so `TestResults/` accumulates the input photo, the generated SVG, and the intermediate debug images alongside it
2. Optionally record notes about what was tested and any observations
3. Bundle with `CreateTestDataArchive` (whole tree, or `-c CATEGORY` for just the relevant case)
4. Edit the auto-created (or existing) `TestResults/VERSION.org` if it needs updating for this batch
5. Share the resulting `dvi3v-TestData-*.tar.xz`

4.2 Using Test Data

1. Extract the archive into a project working directory:

```
tar xJf dvi3v-TestData-*.tar.xz
```

2. Cross-check `TestResults/VERSION.org` against the project version the data was produced with
3. Review the photographs, generated SVGs, and debug images (`-d~/--debug~`'s output; see `man dvi3v`, *Files*, for what each debug image shows) to reproduce, diagnose, or validate against