# 11 Organising Code

## General Advice

It is imperative to keep control over all dependencies of a project. Ideally, all required resources either live inside the project folder (repository) or are maintained via a separate, dedicated mechanism like a package manager.

Otherwise, when the project is opened on a different computer (eg when the Git repository is cloned by another team member), LabVIEW will fail to find those dependencies that are stored in a different (relative) location or not available at all on that machine. Further problems can occur during the build process of such a project.

The preferred location for dependencies is the project folder. For reuse code that should live outside any project, the de-facto standard is `<vi.lib>`, one of the specially treated symbolic locations within LabVIEW, which is maintained via the JKI VI Package Manager. Other special symbolic locations include `<user.lib>` and `<instr.lib>`.

Read more about Paths in LabVIEW.

Any VIs or controls that are loaded from a different path than the ones mentioned above will make LabVIEW link statically to those files, which will cause problems sooner or later.

When using code or VIs from `NI Examples`, **always** make a copy of these example VIs and save it into the project folder.

---

## Project File

See Managing a Project in LabVIEW on ni.com

## Libraries

See Using Libraries in LabVIEW Projects on ni.com

### Performance

Most library types do not load their VIs. They do load their nested libraries. Classes and XControls are the only two types of libraries that load their member VIs.

LabVIEW uses the Read Link Info application method to harvest the linker graph info without loading the files.

### Migrating VIs into .lvlib

*Any callers […] will “accept” the change of the VIs being moved into an LVLIB. This was a behaviour that NI added, intentionally, to ease the migration to LVLIB's (I heard about it from Aristos Queue / SLM). However, callers will not accept a change in a subVI being moved into a different/renamed LVLIB or being removed from an LVLIB. Callers only accept a change of a subVI moving into an LVLIB, as long as the subVI's path on disk is the same.*
(Jim Kring on Discord)

Beware: Migrating VIs into an .lvlib cannot be undone from within LabVIEW! Once you update and save VIs that have the migrated code as dependency, you need a backup or source code control to go back.

## Classes

See LVOOP.

## LLBs

See Creating LLBs on ni.com