6.3 Version Requirements

Go To Top

Return to Prerequisites and Tooling

The team should specify version requirements for each required tool. Version requirements protect Reproducibility and reduce the risk that different developers generate different Artifacts, build different containers, run different runtime behaviour, or collect inconsistent acceptance Evidence.

The handbook defines the version-requirement rules. Section 6.7 defines the version-range notation that checklist instances should use. The applicable tool Baseline checklist instance records the approved product, platform, implementation, acceptable version range, observed version, verification command, verification result, and Evidence reference for a specific phase, build, test run, demonstration, or deployment environment.

The checklist instance should identify version requirements for:

1. Git.
2. Supported shell environment.
3. Repository platform or Repository access tooling, where applicable.
4. DDS vendor, runtime, and toolset.
5. IDL compiler or type-generation tool.
6. Container runtime and container orchestration tool, where applicable.
7. Python and any required Python package-management tooling.
8. Java, C++, JavaScript, TypeScript, or other runtime language tools, where applicable.
9. Build tools and package managers.
10. Test and validation tools.
11. Documentation generation tools.
12. VS Code / Code and recommended extension sets, where applicable.
13. CI/CD or automation runner tooling, where applicable.

The team should prefer bounded ranges when tool behaviour, generated output, runtime compatibility, container behaviour, or acceptance Evidence may change across major versions. The team may use open-ended ranges when the tool has stable backward compatibility or when the checklist instance identifies the associated risk.

The team should avoid vague requirements such as “latest version” unless the tool genuinely supports stable backward compatibility for the Phase 0 workflow. When a tool version affects generated output, runtime behaviour, container builds, validation results, or acceptance Evidence, the checklist instance should pin the expected version or define an acceptable version range.

When a category supports multiple products, platforms, or implementations, each product entry should carry its own version requirement. For example, a DDS checklist instance may record one version range for the primary DDS product and different version ranges for supported alternatives or interoperability targets.
