9.8 Type Compatibility Rules

Go To Top

Return to DDS / IDL / Type Generation

Type compatibility protects communication across Nodes, generated bindings, and DDS implementations. The team should treat IDL changes as controlled Baseline changes because a type change can affect publishers, subscribers, tests, logs, generated code, and interoperability Evidence.

Developers should review IDL changes for compatibility before committing them. A change may affect compatibility when it:

1. Renames a type or field.
2. Changes a field type.
3. Changes key fields.
4. Changes optionality.
5. Removes a field.
6. Changes default assumptions.
7. Changes namespaces or modules.
8. Alters a structure used by an existing topic.

The team should prefer additive, backwards-aware changes where practical. If a change breaks compatibility, the team should:

1. Record the reason.
2. Update the relevant topic definitions.
3. Regenerate affected Artifacts.
4. Update Node implementations.
5. Update tests.
6. Collect new validation Evidence.

When Phase 0 or later phases support multiple DDS products, the compatibility review should include vendor-specific generation and interoperability considerations. The team should not assume that a type definition works across all supported products until the relevant checklist instance and Evidence confirm the tested combinations.

The Topic Catalogue should identify which data structure each topic carries. The Node Catalogue should identify which Nodes publish or subscribe to those topics. The type generation workflow should ensure that generated bindings match those catalogue entries.

The practical rule is simple: an IDL file is not merely a code-generation input. It is a governed exchange contract. The team should deliberately change it, consistently regenerate from it, validate it before use, and preserve Evidence that the generated types support the Phase 0 Baseline.
