15. Logging and Observability This section defines how the Phase 0 baseline uses logging and observability. Logging records what individual scripts, nodes, containers, and tools do. Observability helps the team understand the behaviour of the node network as a whole. Together, they provide the evidence needed to build, run, troubleshoot, review, and accept the Phase 0 baseline. 15.1 Purpose of Logging Logging should make Phase 0 behaviour understandable and reviewable. A developer, tester, reviewer, or demonstration operator should be able to inspect logs and determine what the system attempted to do, what succeeded, what failed, and where to look next. Phase 0 logging should support: Repository setup Prerequisite validation Environment initialisation DDS/IDL type generation Container builds Node startup Node lifecycle changes Control Plane status reporting Control Plane command handling Data Plane publication or subscription behaviour, where applicable Controlled shutdown Error diagnosis Acceptance evidence Logs should explain behaviour without overwhelming the reader. The team should log significant events, decisions, warnings, failures, and lifecycle transitions. The team should avoid noisy logs that obscure important evidence. Logging should not expose secrets, credentials, private keys, access tokens, passwords, sensitive local paths, or confidential data. If a value may be sensitive, the implementation should mask, omit, or summarise it. 15.2 Log Format The team should use a consistent log format so that logs from scripts, nodes, containers, and test tools can be inspected and compared. A consistent format also supports later automation, filtering, dashboards, and evidence extraction. Each log entry should include, where practical: Timestamp Severity level Source component Node identity, where applicable Runtime instance identifier, where applicable Message Event or operation name, where useful Correlation identifier, where useful Error code or exception class, where applicable Evidence or run identifier, where applicable A representative plain-text log entry format is: 2026-06-04T14:32:10Z INFO fx-validation-node lifecycle state=Running message=“Node reached running state” A representative structured log entry format is: {

```
"timestamp": "2026-06-04T14:32:10Z",
"level": "INFO",
"component": "fx-validation-node",
"nodeRole": "ValidationNode",
"event": "LifecycleStateChanged",
"state": "Running",
"message": "Node reached running state"
```

} The team may use plain-text logs for simple Phase 0 scripts and structured logs for nodes or acceptance evidence. If the team uses more than one format, the relevant script, node, or container documentation should identify the format and its purpose. Severity levels should have consistent meanings: DEBUG records detailed diagnostic information for development or troubleshooting. INFO records non-significant events. WARN records unexpected or degraded conditions that do not immediately stop execution. ERROR records failures that prevent a requested operation or node behaviour from completing normally. FATAL records unrecoverable failures that force process termination or make the node unsafe to continue.