Annex C: Requirements

Go to Annexes

Annex C contains the canonical requirements register for this specification.

The requirements are organized according to defined requirement categories. Each requirement has a stable identifier and resides on a separate leaf page so that architecture pages, ConOps activities, realization artifacts, tests, Evidence, and other records can reference the requirement through a stable URI.

The requirements register distinguishes between:

* The governance status of a requirement * The delivery phase associated with a requirement * The source and derivation of a requirement * The architecture and operational concepts associated with a requirement * The ConOps activities associated with a requirement * The verification activities used to determine whether a realization satisfies a requirement

Requirements in this register define required capabilities and constraints without prescribing a particular product, implementation, or realization technology.

Traceability maintains relationships between requirements and specific Reference Architectures, products, architectural capabilities, ConOps activities, or implementations.

Requirement Categories

The requirements register uses the following requirement categories:

* C.1 Mission Objectives * C.2 Operational Requirements * C.3 Functional Requirements

- C.3.1 Test Environment Requirements
- C.3.2 Node Requirements
- C.3.3 Twin Node Requirements
- C.3.4 Virtual Network and Communication Requirements
- C.3.5 State and Time Control Requirements
- C.3.6 Test Definition Requirements
- C.3.7 Test Execution Requirements
- C.3.8 Record and Playback Requirements
- C.3.9 Baseline and Comparison Requirements
- C.3.10 Validation Requirements
- C.3.11 Monitoring and Evidence Requirements
- C.3.12 Catalog and Reuse Requirements

* C.4 Security Requirements * C.5 Logging and Auditing Requirements * C.6 Reliability Requirements * C.7 Performance Requirements * C.8 Maintainability Requirements * C.9 Data Management Requirements * C.10 Interoperability Requirements * C.11 Conformance Requirements

Requirement Identifiers

Each requirement has a stable requirement identifier.

The requirement identifier identifies the requirement independently of its location within the requirements hierarchy. Moving a requirement to another category or namespace does not, by itself, change the requirement identifier.

For example, a Node requirement identified as `NOD-013` remains `NOD-013` if the Node Requirements namespace is relocated within Annex C.

Requirement identifiers SHALL NOT be reused for a different requirement after assignment.

A requirement that is withdrawn, deprecated, or superseded retains its identifier and governance history.

Requirement Decomposition

A source requirement may contain multiple independently verifiable obligations.

Decompose such a requirement into multiple requirements when decomposition improves atomicity, clarity, traceability, or verification.

Derived requirements preserve traceability to the source requirement, architectural concept, model element, or other authoritative basis from which they were derived.

Decomposition SHALL NOT change the approved intent of the source obligation.

A derived requirement does not acquire independent authority merely because it has been decomposed from another statement. Its authority remains traceable to its source and applicable governance process.

Source and Derivation Traceability

Each requirement identifies its authoritative source or derivation basis where applicable.

A requirement may originate from:

* A source requirement * A normative specification or other controlled document * A policy or governed constraint * A conceptual or logical model * An architectural decomposition * An approved governance determination * Another requirement

Source material may be normalized when necessary to produce an atomic, clear, verifiable requirement.

Normalization may correct:

* Terminology * Grammar * Active or passive voice * Normative modality * Ambiguous subjects * Compound obligations * Undefined references * Implementation-specific wording that is inappropriate at the architectural level

Normalization SHALL preserve the source's approved intent.

When a material change in intent is required, the change SHALL be handled through the applicable governance process rather than treated as editorial normalization.

Requirement Page Structure

Each requirement page uses the following sections where applicable:

* Statement * Source * Rationale * Applies To * Verification * Traceability * ConOps Relationship * Delivery Phase * Requirement Status

Additional sections SHALL NOT be added to individual requirement pages merely to duplicate information maintained through backlinks, traceability records, or another authoritative register.

Normative Language

Requirement statements use the following normative terms:

* **SHALL** identifies a mandatory requirement * **SHALL NOT** identifies a mandatory prohibition * **MAY** identifies a permitted capability or action * **SHOULD** identifies a recommendation when advisory language remains appropriate

Normative modal verbs are written in uppercase.

A requirement page may editorially correct source wording for terminology, atomicity, clarity, active voice, normative style, or verification without changing the approved intent of the requirement.

Requirement Quality

Requirements are reviewed for specification quality before approval.

Requirement review examines characteristics including:

* Atomicity * Clear identification of the responsible subject * Defined terminology * Appropriate normative modality * Active voice * Unambiguous scope * Verifiability * Traceability * Consistency with related requirements * Absence of unnecessary implementation constraints * Absence of unresolved conflicts or ambiguous precedence

Specification Defect Analysis (SDA) may identify candidate weaknesses in requirement wording, structure, terminology, traceability, or semantics.

Common Weakness Enumeration (CWE) analysis may classify confirmed specification defects where

applicable.

Automated or AI-assisted analysis may identify, compare, trace, or propose corrections to candidate issues. Automated analysis does not make authoritative governance determinations.

A conflict, ambiguity, or unresolved precedence among governed requirements, policies, specifications, profiles, or other authoritative constraints SHALL be referred to the appropriate governance authority through the applicable Community of Interest (CoI) structure.

Requirement Status

Requirement Status identifies the requirement's governance state.

Possible states include:

* Draft * Proposed * Approved * Deprecated * Superseded * Withdrawn

Requirement Status describes the requirement's authority and lifecycle state. It does not describe whether a particular product or implementation satisfies the requirement.

Realization and Implementation Status

Requirements in this annex are product-neutral and may be realized by one or more architectural capabilities, Reference Architectures, products, services, or implementations.

The canonical requirement does not prescribe a specific realization unless such a constraint is itself part of the approved requirement.

Maintain realization relationships separately through traceability.

For example, requirements in this annex may be realized wholly or partially by:

* Crucible * A conforming DevSecOps Reference Architecture realization * A distributed testing capability * Another DIDO Solutions capability * An external conforming implementation

The applicable realization, rather than the canonical requirement, maintains Implementation Status.

A realization may therefore identify its relationship to a requirement using implementation states such as:

* Implemented * Partially Implemented * Demonstrated * Planned * Unsupported * Not Assessed

Requirement Status and realization-specific Implementation Status remain separate.

Contents

—

Notes for Editors

☐ Existing requirement-family namespaces must be moved into the requirement hierarchy defined under Requirement Categories.

☐ Preserve existing stable requirement identifiers when moving requirement pages.

☐ Update affected backlinks, source-section transclusions, and requirement-family navigation after each namespace migration.

☐ Do not allocate canonical requirements directly to Crucible, DevSecOps, or another realization within the requirement statement. Maintain realization relationships through traceability.

—