

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 requirement page so that architecture pages, ConOps activities, realization artifacts, tests, [Evidence](#), and other records can reference the requirement through a stable URI.

A requirement page may be a leaf page containing an independently verifiable normative obligation or a non-leaf page organizing subordinate requirements.

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](#)

4. [C.4 Security Requirements](#)
5. [C.5 Logging and Auditing Requirements](#)
6. [C.6 Reliability Requirements](#)
7. [C.7 Performance Requirements](#)
8. [C.8 Maintainability Requirements](#)
9. [C.9 Data Management Requirements](#)
10. [C.10 Interoperability Requirements](#)
11. [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 Prefixes

Requirement identifiers use a controlled prefix that identifies the requirement family.

The prefix identifies the subject area of the requirement and remains stable even if the requirement is moved within the Annex C hierarchy.

Prefix	Requirement Category
MO	Mission Objective
OPR	Operational Requirement
ENV	Test Environment Requirement
NOD	Node Requirement
TWIN	Twin Node Requirement
NET	Virtual Network and Communication Requirement
STC	State and Time Control Requirement
TDF	Test Definition Requirement
TEX	Test Execution Requirement
RPB	Record and Playback Requirement
BLC	Baseline and Comparison Requirement
VAL	Validation Requirement
EVD	Monitoring and Evidence Requirement

Prefix	Requirement Category
CAT	Catalog and Reuse Requirement
SEC	Security Requirement
AUD	Logging and Auditing Requirement
REL	Reliability Requirement
PER	Performance Requirement
MNT	Maintainability Requirement
DAT	Data Management Requirement
INT	Interoperability Requirement
CON	Conformance Requirement

Requirement identifiers use the form:

PREFIX-NNN

where:

- PREFIX identifies the requirement family
- NNN is a sequential identifier within that family

For example:

ENV-001
NOD-013
TEX-004
SEC-002

A requirement prefix SHALL NOT change solely because the requirement page or requirement category is reorganized within Annex C.

A retired or superseded requirement identifier SHALL NOT be reassigned to a different requirement.

Requirement Decomposition

A source requirement may contain multiple independently verifiable obligations.

A source requirement containing multiple independently verifiable obligations may be decomposed 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.

The applicable governance process SHALL approve a material change in requirement intent.

Editorial normalization SHALL NOT introduce a material change in intent.

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

A leaf requirement page contains an independently verifiable normative obligation and therefore includes a **Statement** section and a **Verification** section.

A non-leaf requirement page organizes subordinate requirements and does not establish a separate conformance obligation unless explicitly stated otherwise. A non-leaf requirement page therefore normally omits **Statement** and **Verification** sections.

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 Discipline and Authoring (SDA) identifies candidate weaknesses in requirement wording, structure, terminology, traceability, or semantics.

The Common Specification Weakness Enumeration (CWE) evaluates candidate findings, confirms applicable defects, and records their classification, severity, governing rule, and

applicable defect pattern.

Automated or AI-assisted analysis may identify, compare, trace, or propose corrections to candidate issues.

Automated or AI-assisted analysis SHALL NOT resolve conflicts, ambiguities, or precedence among governed requirements, policies, specifications, profiles, or other authoritative constraints.

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 forms part of the approved requirement.

Traceability records maintain relationships between canonical requirements and their realizations.

The applicable realization maintains Implementation Status. Implementation Status is not a property of the canonical requirement.

A realization may 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

- [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](#)
- 4. [C.4 Security Requirements](#)
- 5. [C.5 Logging and Auditing Requirements](#)
- 6. [C.6 Reliability Requirements](#)
- 7. [C.7 Performance Requirements](#)
- 8. [C.8 Maintainability Requirements](#)
- 9. [C.9 Data Management Requirements](#)
- 10. [C.10 Interoperability Requirements](#)
- 11. [C.11 Conformance Requirements](#)
 - [C.1 Mission Objectives](#)
 - [C.2 Operational Requirements](#)
 - [C.3 Functional Requirements](#)
 - [C.11 Conformance Requirements](#)

Notes for Editors

☐ Remove the explicit creation links under Contents after all planned requirement-category pages have been created and the indexmenu discovers them.

Preserve stable requirement identifiers when moving or reorganizing requirement pages.

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

© 2026 Dido Solutions, Inc. and Jackrabbit Consulting, Inc.

From:
<https://wiki.didosolutions.com/> - **Dido Solutions, Inc Wiki**

Permanent link:
<https://wiki.didosolutions.com/dido/03-dido-te/99-annexes/annex-c-requirements/start>

Last update: **2026/08/19 18:16**

