Annex C: Requirements

Go to Annexes

Annex C contains the canonical requirements register for the Distributed Immutable Data Object Test Environment (DIDO-TE).

Annex C organises requirements by requirement class. Each normative requirement receives a stable identifier and resides on a separate wiki page. Architecture pages, implementation artifacts, tests, Evidence, verification records, and other records reference each requirement through its stable identifier and URI.

The draft Requirements Register provides one source of requirements and supporting information for this annex. The identifiers `REQ-0001` through `REQ-0091` identify requirements in that source register. Annex C preserves those identifiers for traceability but assigns canonical DIDO-TE identifiers to requirements derived from the register.

Additional requirements may derive from the references identified in Annex B: References, including:

- [DTE1] U.S. Patent Application US20220237111A1
- [DTE2] Non-Traditional BAA Submission
- [DTE3] Draft BAA White Paper
- [DTE4] DIDO Reference Data Model

DIDO-TE requirements conform to the:

- Common Specification Weakness Enumeration (CWE)
- OMG Specification Discipline and Authoring (SDA)
- DIDO Solutions shared Terms and Definitions

SDA provides the detection layer for candidate specification weaknesses. CWE provides the authoritative evaluation layer for confirming defects, identifying governing rules, assigning classifications and severity, selecting applicable defect patterns, and recording traceable findings.

Annex C distinguishes among:

- The canonical DIDO-TE requirement
- The source material
- The requirement lineage
- The requirement class
- The governance status of the requirement
- The delivery phase associated with the requirement
- The implementation status of DIDO-TE relative to the requirement
- The verification activities used to determine whether DIDO-TE satisfies the requirement
- The Evidence required to support the verification result
- The architecture, ConOps, and operational concepts that support the requirement

Old Requirement Categories

New Requirements

Annex C: Requirements

Go to Annexes

This annex identifies the canonical requirement records for the DIDO Test Environment (DIDO-TE).

Requirements are organised by architectural concern rather than by document chapter. Each requirement is maintained as an individual wiki page with a stable identifier and URI. Annex D provides traceability from these requirements to the body of this specification and to their authoritative source documents.

Requirement Categories

- C.1 Mission Objectives
- C.2 Operational Requirements
- C.3 Functional Requirements
- C.11 Conformance Requirements

Requirement Identifiers

DIDO-TE uses the following requirement-class identifiers:

Requirement Class	Prefix
Mission Objective	MO
Operational Requirement	OR
Functional Requirement	FR
Security Requirement	SEC
Logging and Auditing Requirement	LOG
Reliability Requirement	REL
Performance Requirement	PER
Maintainability Requirement	MNT
Data Management Requirement	DAT
Interoperability Requirement	INT
Future Capability Requirement	FUT
Key Value Proposition Requirement	KVP

Each requirement identifier consists of the requirement-class prefix followed by a three-digit number.

Examples include:

- `MO-001`
- `OR-001`
- `FR-001`
- `SEC-001`
- `PER-001`

A requirement identifier remains stable when the requirement moves to another section or undergoes an editorial revision that does not substantively change the requirement.

Annex C does not reassign a retired, withdrawn, deprecated, or superseded identifier to another requirement.

Requirement Decomposition

Each normative requirement expresses one independently interpretable and independently verifiable obligation.

When a source requirement contains multiple normative obligations, Annex C preserves the composite requirement on a non-leaf parent page and assigns a lowercase suffix to each derived leaf requirement.

For example:

- `FR-040` identifies the composite requirement represented by the parent page
- `FR-040a` identifies the first derived requirement
- `FR-040b` identifies the second derived requirement

The non-leaf parent page retains a trailing `:start` in its namespace. Each derived leaf requirement omits a trailing `:start`.

The parent page preserves the composite source statement and links to each derived leaf requirement. Each derived requirement identifies the parent requirement and the specific obligation preserved from the composite statement.

Decomposition does not change the source provenance. A derived leaf requirement retains traceability to both its parent requirement and the external source material from which the requirement originated.

Source and Derivation Traceability

The **Source** section identifies the external source material supporting a requirement.

Sources include:

- The draft Requirements Register
- Annex B references
- Approved architecture content
- Approved ConOps content
- Approved governance decisions
- Other authoritative source artifacts

A citation identifies the most precise available source location, such as a requirement identifier, section, clause, claim, page, figure, table, diagram, package, class, datatype, attribute,

association, or model element.

For example:

```
[[dido:03-dido-te:99-annexes:annex-b-references:dte-002|[DTE2]]],
Section 2.1.1, Key Tasks.
```

The **Derived From** section identifies requirement lineage within Annex C. It identifies the non-leaf parent requirement and the particular source obligation preserved by the derived leaf requirement.

For example:

```
[[dido:03-dido-te:99-annexes:annex-c-requirements:02-functional-
requirements:02-06-test-definition:fr-040:start|FR-040]], source
obligation concerning test-step sequencing.
```

The **Source** and **Derived From** sections serve different purposes and are not mutually exclusive. A decomposed requirement normally uses both:

- **Source** records external provenance.
- **Derived From** records requirement decomposition and lineage.

A requirement that does not result from decomposition omits the **Derived From** section.

Conceptual-Model Derivation

[DTE4] records a historical Conceptual Model. Requirements derived from `[DTE4]` identify the precise package, diagram, model element, datatype, attribute, association, enumeration, or qualified model path supporting the requirement.

For example:

```
[[dido:03-dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]],
conceptual datatype //yes_no_type//.
```

DIDO-TE preserves the following model progression:

1. The Conceptual Model defines implementation-independent concepts, meanings, relationships, and semantic value spaces.
2. A Logical Model maps applicable conceptual elements and datatypes to implementable logical representations.
3. A Physical Model maps applicable logical representations to concrete platform-specific structures, datatypes, encodings, and constraints.

A requirement derived from a conceptual-model element preserves the implementation-independent meaning of the source element. It does not prescribe a logical or physical representation unless an approved constraint expressly requires that representation.

Conceptual-to-logical and logical-to-physical mappings remain explicit and traceable. Each

mapping identifies:

- The source element
- The target element
- The value correspondence
- Applicable constraints
- Null semantics, when applicable
- Any information loss or semantic change
- The rationale for the selected mapping

Source Normalisation

Annex C corrects spelling, spacing, capitalisation, and naming inconsistencies when incorporating historical source material into current DIDO-TE requirements.

A derivation record preserves the original source expression and identifies the normalised expression when the correction affects traceability.

For example:

Source name in [DTE4]	**Normalised name**
`Parameter_Diretion_Type`	`Parameter Direction Type`
`ParameterLlist Type`	`Parameter List Type`
`Parameter_Definition_Type`	`Parameter Definition Type`

An editorial normalisation preserves the original meaning.

A change to values, constraints, cardinality, relationships, or meaning constitutes a substantive model change. Annex C records the rationale and disposition for each substantive change.

Requirement Page Structure

Each leaf requirement page uses the following sections where applicable:

- Statement
- Source
- Derived From
- Rationale
- Applies To
- Verification
- Referenced By
- Delivery Phase
- Implementation Status
- Requirement Status
- Issues
- Notes for Editors

Every leaf requirement contains a **Source** section.

A leaf requirement contains a **Derived From** section when it results from the decomposition of a non-leaf parent requirement.

The **Statement** section contains only the normative requirement.

The **Verification** section identifies objective activities and results that determine whether DIDO-TE satisfies the requirement.

The **Referenced By** section uses backlinks to identify pages that reference the requirement.

Normative Language

Requirement Statements use the following normative terms:

- **SHALL** identifies a mandatory requirement
- **SHALL NOT** identifies a mandatory prohibition
- **MAY** identifies a permission
- **SHOULD** identifies an advisory recommendation
- **SHOULD NOT** identifies an advisory prohibition

Only **SHALL** and **SHALL NOT** establish mandatory conformance obligations.

The Statement section contains only the normative statement.

Rationale, examples, implementation information, verification information, source information, derivation information, and editorial guidance remain outside the Statement section.

Requirement Quality

Each DIDO-TE requirement conforms to the applicable CWE and SDA rules.

Each requirement:

- Has a unique and stable identifier
- Expresses a single, independently interpretable obligation
- Identifies the responsible actor
- Identifies the required behaviour, outcome, permission, or prohibition
- Uses defined terminology consistently
- Uses explicit normative language
- Avoids vague, weak, open-ended, and logically ambiguous expressions
- Avoids implementation-specific mechanisms unless an approved constraint requires the mechanism
- Supports objective verification
- Preserves traceability to source material
- Preserves traceability to architecture, implementation, tests, Evidence, and conformance artifacts

SDA analysis identifies candidate weaknesses. CWE evaluation confirms defects and records:

- The governing rule
- The defect classification
- The assigned severity
- The applicable defect pattern
- The affected requirement
- The supporting rationale
- The required disposition

Requirement Status

Requirement Status identifies the governance state of the requirement.

Possible states include:

- Draft
- Proposed
- Approved
- Deprecated
- Superseded
- Withdrawn

Implementation Status

Implementation Status identifies the state of DIDO-TE relative to the requirement.

Possible states include:

- Implemented
- Partially Implemented
- Demonstrated
- Planned
- Deferred
- Strategic Unscheduled
- Unsupported
- Not Assessed

Requirement Status and Implementation Status remain separate.

An approved requirement may remain planned, partially implemented, deferred, unsupported, or not assessed.

Contents

Notes for Editors

Annex C is the authoritative source for DIDO-TE normative requirements.

The draft Requirements Register remains source material. Its `REQ-0001` through `REQ-0091` identifiers remain source-reference identifiers and do not become canonical DIDO-TE requirement identifiers.

Treat [DTE4] as a historical Conceptual Model and a source of conceptual-model provenance. Do not treat every class, datatype, relationship, or diagram in `[DTE4]` as a current normative DIDO-TE requirement.

Identify the precise source location for every requirement. Do not cite only a reference identifier when a more precise location exists.

Use **Source** to record external provenance. Use **Derived From** to record requirement decomposition and lineage. Do not use one section as a substitute for the other.

Preserve original source identifiers, names, and expressions in traceability records.

Correct spelling and naming defects in current DIDO-TE requirements. Record the original and normalised expressions when a correction affects traceability.

Do not classify changes to values, constraints, cardinality, relationships, or meaning as editorial corrections.

Preserve the distinction among Conceptual Models, Logical Models, and Physical Models.

Do not place implementation-specific datatypes, encodings, products, or technologies in a conceptual-model requirement unless an approved constraint expressly requires them.

Record conceptual-to-logical and logical-to-physical mappings explicitly and preserve required semantic distinctions through each mapping.

Apply SDA pattern detection before performing CWE evaluation.

Preserve traceability from each DIDO-TE requirement to:

- External source material
- Its parent requirement, when decomposition applies
- Applicable ConOps scenarios
- Applicable architecture sections
- Implementation artifacts
- Verification activities
- Verification results
- Evidence
- Identified issues

Use a non-leaf parent page only when requirement decomposition requires one.

Leaf requirement pages omit a trailing `:start` from their namespaces.

Non-leaf requirement pages retain a trailing `:start` in their namespaces.

Do not change an assigned requirement identifier because the requirement moves to another section.

Do not reuse an identifier assigned to a retired, withdrawn, deprecated, or superseded requirement.

Do not rename a requirement page after an external citation unless a redirect or move plan is in place.
