[DTE4] DIDO Reference Data Model

Go to Annex B: References

Reference

DIDO Solutions, Inc. *DIDO Reference Data Model*. Native MagicDraw UML model. MagicDraw UML 19.0 v9. 30 December 2019.

Bibliographic Information

Attribute	Value
Reference Identifier	[DTE4]
Title	DIDO Reference Data Model
Filename	`DIDO Reference Data Model.mdzip`
Artifact Type	Native UML model
Model Level	Conceptual Model
Modelling Tool	MagicDraw UML
Tool Version	19.0 v9
Model Date	30 December 2019
Preparing Organisation	DIDO Solutions, Inc.
File Format	MagicDraw compressed model package (`.mdzip`)
Diagram Count	127
UML Class Diagrams	78
SQL Diagrams	39
Other Diagrams	10
Status	Historical conceptual model

Model Scope

The DIDO Reference Data Model defines conceptual entities, semantic datatypes, attributes, associations, packages, and diagrams for Distributed Immutable Data Object environments.

The model covers subject areas associated with:

- DIDO Test Environments
- Test Plans
- Test Sets
- Test Cases
- Tests
- Test Steps
- Test Executables
- Expected Results
- Test Results

- Test Runs
- Test Tags
- Runtime Environments
- Nodes and Node Sets
- Node Profiles
- Node Classifications
- Composite Applications
- Application Containers
- DIDO Platforms
- Domains
- Ecosystems
- Ecospheres
- Governance
- Users and contact information
- Glossary and supporting reference information

The model extends beyond DIDO-TE. Cite only the packages, diagrams, and model elements relevant to the applicable DIDO-TE statement or requirement.

DIDO-TE Model Content

The model contains native, editable diagrams and model elements relevant to DIDO-TE, including:

- *Test Plan Package*
- *Test Environment Package*
- *Test Set*
- *Test Case*
- *Test Plan*
- *Test*
- *Test Step*
- *Test Executable*
- *Expected Results*
- *Test Result*
- *Test Run*
- *Test Tag*
- *Runtime Environment Package*
- *Node Set*
- *Node Profile*
- *Node Classification*
- *Composite Application*
- *Application Container*
- *DIDO Platform*
- *Runtime Environment*
- *Domain Package*

The native package retains the model elements, attributes, associations, diagram definitions, and layout information used to produce these diagrams.

Datatypes

The model defines implementation-independent semantic datatypes. The model contains:

- 26 UML `DataType` elements
- 8 UML `Enumeration` elements
- 4 UML `PrimitiveType` elements

Examples include:

- `dido_id_type`
- `dido_name_type`
- `dido_description_type`
- `dido_date_type`
- `dido_blob_type`
- `dido_key_type`
- `document_type`
- `domain_namespace_type`
- `namespace_type`
- `uri_type`
- `version_type`
- `release_number_type`
- `release_status_type`
- `email_address_type`
- `filename_type`
- `node_count_type`
- `cpu_counter_type`
- `mebibyte_type`
- `gibibyte_type`
- `yes_no_type`
- `Parameter_Type`
- `ParameterLlist Type`
- `Parameter_Diretion_Type`
- `Parameter_Definition_Type`
- `Verdict_Type`
- `JSON String Type`

Some enumerations define explicit semantic value spaces. Examples include:

Conceptual datatype	**Defined values**
`yes_no_type`	`Unknown`, `Yes`, `No`
`Parameter_Diretion_Type`	`Unknown`, `In`, `Out`, `InOut`
`Verdict_Type`	Verdict values including `None`, `Pass`, and `Inconclusive`
`Parameter_Definition_Type`	Values including `Any`, `Boolean`, `Composite`, `Date`, `Enumeration`, `External`, `Integer`, `Real`, `String`, `UUID`, and `XHTML`

These conceptual datatypes define meaning and semantic distinctions. They do not prescribe a database datatype, programming-language type, storage representation, wire encoding, or platform product.

Model-Level Mappings

DIDO-TE preserves the following 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.

One conceptual datatype may support multiple logical mappings. One logical datatype may support multiple physical mappings.

For example, the conceptual datatype `yes_no_type` defines three semantic values:

- `Unknown`
- `Yes`
- `No`

A Logical Model might map these values to a nullable Boolean:

Conceptual value	Logical value
`Unknown`	`NULL`
`Yes`	`TRUE`
`No`	`FALSE`

Another Logical Model might map the same values to a signed integer:

Conceptual value	Logical value
`Unknown`	`0`
`Yes`	`+1`
`No`	`-1`

A Physical Model then maps the selected logical representation to a platform-specific datatype and defines its encoding, null semantics, constraints, and validation rules.

Each mapping preserves the semantic distinctions required by the conceptual datatype or explicitly records any information loss or semantic change.

Relevance to DIDO-TE

This reference provides conceptual-model provenance for DIDO-TE entities, datatypes, associations, packages, and diagrams.

It provides source material associated with:

- Test environment composition
- Test planning
- Test definition
- Test execution
- Test steps
- Test executables
- Expected results
- Test results
- Test runs
- Node configuration
- Node classification
- Runtime environments
- Application composition
- Platform descriptions
- Conceptual datatypes
- Conceptual-to-logical model mappings
- Logical-to-physical model mappings

Use `[DTE4]` to establish conceptual provenance. Do not treat a class, datatype, association, diagram, or other historical model element as a current normative DIDO-TE requirement unless Annex C derives, normalises, traces, reviews, and approves the requirement explicitly.

Citation Use

DIDO-TE pages cite this reference using:

[DTE4]

A citation identifies the applicable package, diagram, class, datatype, attribute, association, enumeration, or other model element.

For a diagram:

```
[[dido:03-dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]], UML Class Diagram, //Test Environment Package//.
```

For a class or other named model element:

```
[[dido:03-dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]], model element //Test Plan//.
```

For a conceptual datatype:

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

For an attribute or association, identify its owning model element:

```
[[dido:03-dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]], model
```

```
element //Test Case//, attribute //Attribute Name//.
```

If duplicate names occur, include the containing package or qualified model path.

Exported Diagrams

Diagrams exported from `[DTE4]` serve as documentation figures or model views. An exported diagram does not constitute a separate Annex B reference unless it possesses independent authority, provenance, or revision control.

A figure caption identifies `[DTE4]` as the source.

For example:

```
Figure X: Test Environment Package class diagram exported from [[dido:03-
dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]].
```

When documentation modifies an exported diagram, its caption identifies the figure as an adaptation and describes material changes.

For example:

```
Figure X: DIDO-TE Test Environment conceptual model adapted from [[dido:03-
dido-te:99-annexes:annex-b-references:dte-004|[DTE4]]].
```

Terminology Normalisation

The historical model contains spelling errors and inconsistent naming conventions. Current DIDO-TE requirements and architecture content use corrected and normalised names.

Known corrections include:

Source name in [DTE4]	Normalised name
Parameter_Diretion_Type	Parameter Direction Type
ParameterLlist Type	Parameter List Type
Parameter_Definition_Type	Parameter Definition Type
JSON String Type	JSON String Type

A derivation record preserves both the source name and the normalised name when a correction affects traceability.

A correction to spelling, spacing, capitalisation, or naming style constitutes an editorial normalisation when it preserves the original meaning. A change to values, relationships, constraints, cardinality, or semantics constitutes a model change and requires an explicit rationale and disposition.

Related Model Files

The following model packages share DIDO model content or lineage with `[DTE4]`:

- `DIDO - RI - Data Model.mdzip`
- `DIDO - RI - Data Model Conceptual Model.mdzip`

These files do not constitute byte-for-byte copies of `[DTE4]`. Their internal model content, file inventories, and model-file hashes differ.

Treat them as related versions or variants until model governance establishes their exact revision, derivation, or supersession relationships.

Do not create separate Annex B references for these files unless they possess independent authority or contain material used directly as the source of a DIDO-TE requirement.

Relationship to Other References

[DTE4] provides the native conceptual model associated with concepts described or applied in other DIDO-TE references.

See:

- [DTE1] U.S. Patent Application US20220237111A1
- [DTE2] Non-Traditional BAA Submission
- [DTE3] Draft BAA White Paper

[DTE1] records the patented subject matter. [DTE2] and [DTE3] describe proposed applications of DIDO-TE. [DTE4] supplies a native conceptual model containing relevant entities, datatypes, relationships, and diagrams.

Cite the source that directly supports the applicable statement or requirement derivation.

Status

[DTE4] records a historical conceptual model dated 30 December 2019.

The model establishes provenance but does not, by itself, establish the current normative DIDO-TE architecture. Current DIDO-TE work reviews the model content, corrects editorial defects, resolves inconsistencies, and derives explicit requirements through the Annex C requirements process.

The model date, tool version, and historical names remain part of the source record.

Notes for Editors

Preserve `[DTE4]` and this page namespace as stable citation identifiers.

Preserve the native `.mdzip` file as the canonical source artifact. Do not substitute exported images for the native model.

Identify the precise package, diagram, class, datatype, attribute, association, enumeration, or model path when deriving a DIDO-TE requirement from `[DTE4]`.

Treat `[DTE4]` as a Conceptual Model. Do not infer logical or physical representations from conceptual datatype names.

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

Record conceptual-to-logical and logical-to-physical mappings explicitly. Each mapping identifies its source element, target element, value correspondence, constraints, and any information loss or semantic change.

Correct spelling and naming defects in current DIDO-TE content. Preserve the original source expression in traceability records.

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

Treat exported diagrams as figures or views derived from `[DTE4]`, not as independent references, unless an export possesses independent authority, provenance, or revision control.

Do not assume every package or diagram in `[DTE4]` belongs within the current DIDO-TE scope. The model contains broader DIDO subject matter.

Record the revision, provenance, and supersession relationship before replacing `[DTE4]` with another model package.
