5. Common Information Types

Go to DIDO-TE Cover Page

Common Information Types define reusable conceptual value types used by multiple elements of the DIDO-TE Information Model.

These types are defined independently of the classes, attributes, associations, or other information-model elements that use them. A Common Information Type may therefore be referenced by multiple model elements without redefining its conceptual meaning or value structure at each point of use.

The initial Common Information Types are derived from [DTE4] DIDO Reference Data Model and are normalized where necessary for use by DIDO-TE.

Purpose

Common Information Types provide reusable conceptual definitions for information such as:

- identifiers and keys;
- names and descriptions;
- binary content;
- dates, times, and durations;
- versions and release numbers;
- namespaces and URIs;
- file and document references;
- sizes, counts, and other quantified values;
- parameters and parameter values;
- controlled Enumerations;
- status values;
- test verdicts; and
- other values referenced by DIDO-TE Information Model elements.

Defining these types independently allows the Information Model to distinguish between:

- the conceptual semantics and value structure of a reusable type; and
- the semantic role of an attribute that uses that type.

For example, a Test, Test Plan, Test Set, Test Case, and Test Step may each have an identifier. Those attributes may use the same **DIDO ID Type** while retaining different semantic roles within their respective classes.

Relationship to the Information Model

Section 6, DIDO-TE Information Model defines the classes, attributes, associations, and multiplicities that describe DIDO-TE information.

A **Common Information Type** defines the conceptual value semantics of an attribute independently

of the class in which that attribute occurs.

An **Information Model Class** provides the context in which information is described. The class defines one or more attributes, and each attribute identifies the semantic role of a value within that class.

An **Information Model Attribute** references a Common Information Type to identify the conceptual kind, value space, units, structure, or other value semantics applicable to that attribute.

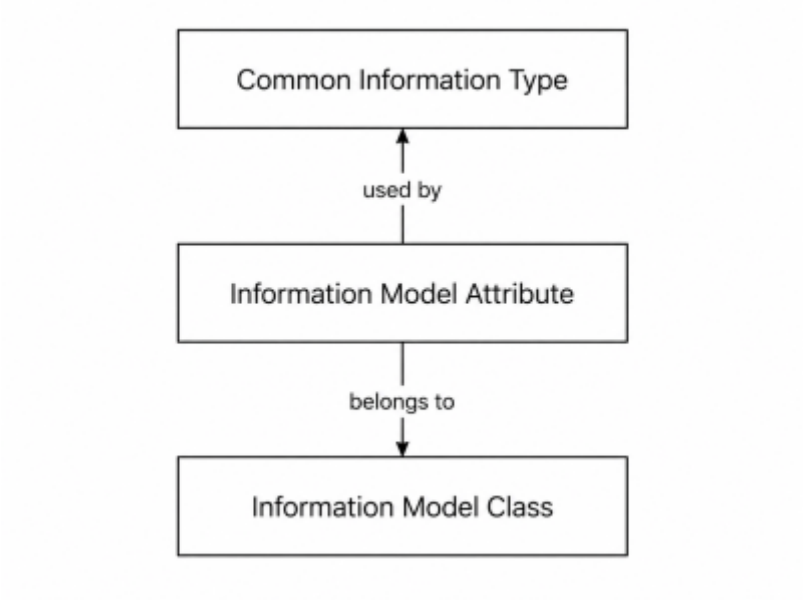

Figure 5-1: Relationship of Common Information Types to Information Model Classes and Attributes

The relationship shown in Figure 5.1-1 can be summarized as follows:

Element	Responsibility
Information Model Class	Establishes the context in which the information exists.
Information Model Attribute	Defines the semantic role of the information within the class.
Common Information Type	Defines the conceptual semantics and value characteristics of the information represented by the attribute.

For example, a **Test** may define an attribute named **Test ID**. The Test class establishes the context, **Test ID** defines the semantic role of the attribute, and **dido_id_type** defines the conceptual value semantics of the identifier.

The same Common Information Type may be referenced by attributes in multiple Information Model classes. Each attribute retains its own semantic role while sharing the conceptual value semantics defined by the Common Information Type.

A Common Information Type retains its conceptual semantics independently of the class or attribute that uses it.

Model Levels

DIDO-TE distinguishes among three model levels:

- **Conceptual Model** — defines the meaning, value space, structure, units, and conceptual constraints of information;
- **Logical Model** — defines how conceptual information is represented using a particular logical type system, language, notation, or interchange technology; and
- **Physical Model** — defines the concrete representation used by a particular implementation, platform, storage system, runtime, or transport.

Section 5 defines **Conceptual Model** information types only.

Logical mappings to technologies such as SQL, JSON, XML, HTML, OMG IDL, Java, ECMAScript, Rust, or other type systems are defined separately.

Physical platform representations are also defined separately.

A logical or physical representation does not redefine the semantics of its conceptual type.

Contents

- 5.1 Conceptual Information Type Model
- 5.2 Common Information Type Catalog

Source

- [DTE4] DIDO Reference Data Model, conceptual datatypes, Enumerations, Primitive Types, and related conceptual model elements.

Notes for Editors

- Maintain Common Information Types independently of the Information Model classes that use them.
- Section 5 defines conceptual information types only.
- Define logical mappings separately for SQL, JSON, XML, HTML, OMG IDL, Java, ECMAScript, Rust, and other applicable logical type systems.
- Define physical platform representations separately from both the conceptual and logical models.
- Do not define a conceptual type by a programming-language, database, serialization, storage, or wire-format realization.