INTERNATIONAL STANDARD

ISO 21597-1

First edition
2020-04

Information container for linked document delivery — Exchange specification —

Part 1: Container

Conteneur d'informations pour la livraison de documents liés — Spécification d'échange —

Partie 1: Conteneur

Reference number
ISO 21597-1:2020(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 59, *Buildings and civil engineering works*, Subcommittee SC 13, *Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM)*, in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/TC 442, *Building Information Modelling (BIM)*, in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

A list of all parts in the ISO 21597 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 21597 series has been developed in response to a recognized need within the construction industry to be able to handle multiple documents as one information delivery.

Information deliveries are often a combination of drawings, information models (representing built or natural assets in the physical world), text documents, spreadsheets, photos, videos, audiofiles, etc. Increasingly, this may also include datasets based on any ontology. An ability to specify relationships using links between information elements in those separate documents can contribute significantly to the value of an information delivery. The composition of such a package arises both from the requirements of the process, e.g. delivery of as-built information, and from the specific functional purpose e.g. performing a quantity take-off or communication about issues in 3D models.

In this document a specification is given for a container that stores documents, along with a means of linking otherwise disconnected data within those documents.

The container format includes a header file and optional link files that define relationships by including references to the documents, or to elements within them. The header file uniquely identifies the container and its contractual or collaborative intention. This information is defined using the RDF, RDFS and OWL semantic web standards.

The header file, along with any additional RDF(S)/OWL files or resources, forms a suite that may be directly queried by software. The link references may be interpreted by the recipient applications or reviewed interactively by the recipient. Where it includes link references into the content of documents that don't support standardized querying mechanisms, their resolution may depend on third party interpreters.

The format can also be used to deliver multiple versions of the same document.

Information container for linked document delivery — Exchange specification —

Part 1: Container

IMPORTANT — The electronic file of this document contains colours which are considered to be useful for the correct understanding of the document. Users should therefore consider printing this document using a colour printer.

1 Scope

This document defines an open and stable container format to exchange files of a heterogeneous nature to deliver, store and archive documents that describe an asset throughout its entire lifecycle.

It is suitable for all parties dealing with information concerning the built environment, where there is a need to exchange multiple documents and their interrelationships, either as part of the process or as contracted deliverables. The format is intended to use resources either included in the container (such as documents) or referenced remotely (such as web resources). A key feature is that the container can include information about the relationships between the documents. Relevant use-cases reflect the need for information exchange during the entire life cycle of any built asset and can include, but are not limited to, the handover of

1. a published bidding package,
2. required project deliverables at a specific project stage (e.g. when proposing different design scenarios),
3. shared information as background or for further development,
4. published approval packages, or
5. information about versions between partners to provide a means to reference particular states of the information and track changes.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 21320-1, *Information technology — Document Container File — Part 1: Core.*

IANA. Internet Assigned Numbers Authority *Media Types*. [viewed 6 May 2019]. Available from: https://www.iana.org/assignments/media-types/media-types.xhtml

W3C-OWL2-SPEC. Motik B., Patel-Schneider P.F., Parsia B. eds. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation, 11 December 2012 [viewed July 22nd 2019]. Latest version available at http://www.w3.org/TR/owl2-syntax/

W3C-RDF11-CONCEPTS. Cyganiak R., Wood D., Lanthaler M. *RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation*, 25 February 2014 [viewed July 22nd 2019]. Latest version available at http://www.w3.org/TR/rdf11-concepts/

W3C-RDF11-SCHEMA. Brickley D., Guha R.V. *RDF Schema 1.1*. W3C Recommendation, 25 February 2014 [viewed July 22nd 2019]. Latest version available at http://www.w3.org/TR/rdf-schema/

W3C-RDF11-XML. Gandon F., Schreiber G. *RDF 1.1 XML Syntax*. W3C Recommendation, 25 February 2014 [viewed July 22nd 2019]. Latest version available at http://www.w3.org/TR/rdf-syntax-grammar/

W3C-XML-DATATYPES. Peterson D., Gao S., Malhotra A., Sperberg-McQueen C.M., Thompson H.S. eds. (Version 1.1) and Biron P.V., Malhotra A. eds. (Version 1.0). *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. W3C Recommendation, 5 April 2012 [viewed July 22nd 2019]. Latest version available at http://www.w3.org/TR/xmlschema11-2/

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1.1
container
file that conforms to the ISO 21597 series

3.1.2
payload
primary information in the form of *documents* (3.1.3) that is included within the *container* (3.1.1)

Note 1 to entry: This does not include the header file (Index.rdf) or the *ontology* (3.1.7) *resource* (3.1.14) files.

3.1.3
document
fixed and structured amount of information that can be managed and interchanged as a unit between users and systems

Note 1 to entry: This unit may not necessarily be human perceptible. Information is usually stored on a data medium.

Note 2 to entry: Used in the ISO 21597 series to refer to any document that forms part of the *payload* (3.1.2) in the container, including any 2D or 3D models that represent built or natural assets in the physical world; these may be held in any standard or proprietary format.

3.1.4
internal document
document (3.1.3) located within the *container* (3.1.1)

3.1.5
external document
document (3.1.3) located outside the *container* (3.1.1)

3.1.6
link
relation between *documents* (3.1.3), including between elements in documents

3.1.7
ontology
specification of concrete or abstract things, and the relationships among them, in a prescribed domain of knowledge

Note 1 to entry: The specification should be computer processable.

Note 2 to entry: The definition is adapted from W3C-OWL2-SPEC.

3.1.8
container ontology
RDF(S)/OWL file providing the *object* (3.1.23) *classes* (3.1.15) and properties that shall be used to specify the contents of a *container* (3.1.1)

3.1.9
linkset ontology
RDF(S)/OWL file providing the *object* (3.1.23) *classes* (3.1.15) and properties that shall be used to specify *links* (3.1.6) between *documents* (3.1.3) in a *container* (3.1.1)

3.1.10
dataset
RDF(S)/OWL file that contains *individuals* (3.1.16) that comply with the *classes* (3.1.15) as specified by *ontologies* (3.1.7)

3.1.11
index dataset
RDF(S)/OWL file containing an index of the contents of the *container* (3.1.1)

3.1.12
link dataset
RDF(S)/OWL file containing *links* (3.1.6) as defined in the ISO 21597 series

3.1.13
serialisation
encoding of an *ontology* (3.1.7) or *dataset* (3.1.10) into a format that can be stored, typically in a file

Note 1 to entry: The definition is adapted from W3C-RDF11-XML.

3.1.14
resource
something in the world (the "universe of discourse") denoted by an IRI or literal

Note 1 to entry: Anything can be a resource, including physical things, *documents* (3.1.3), abstract concepts, numbers and strings; the term is synonymous with "entity" as it is used in the RDF Semantics specification.

Note 2 to entry: The definition is adapted from W3C-RDF11-CONCEPTS.

3.1.15
class
set of *individuals* (3.1.16) having the same characteristics

Note 1 to entry: The definition is adapted from W3C-RDF11-SCHEMA, 2.2.

3.1.16
individual
resource (3.1.14) that has been placed into any RDFS *class* (3.1.15) as an instance of that class

Note 1 to entry: Like RDF classes, every OWL class is associated with a set of individuals, called the class extension; the individuals in the class extension are the instances of the class.

Note 2 to entry: There are two types of individuals in the syntax of OWL 2. Named individuals are given an explicit name that can be used in any *ontology* (3.1.7) to refer to the same *object* (3.1.23). Anonymous individuals do not have a global name and are thus local to the ontology in which they are contained.

Note 3 to entry: The definition is adapted from W3C-OWL2-SPEC, 5.6.

3.1.17
object property
OWL property that links *individuals* (3.1.16) to other individuals

Note 1 to entry: The definition is adapted from W3C-OWL2-SPEC, 5.3.

3.1.18
datatype property
OWL property that can relate *individuals* (3.1.16) to literals

Note 1 to entry: Literals can be strings, numbers, date types, etc.

Note 2 to entry: The definition is adapted from W3C-OWL2-SPEC, 5.4.

3.1.19
namespace
group of identifiers for elements and attributes that are collectively bound to a URI such that their use will not cause naming conflicts

Note 1 to entry: The definition is adapted from W3C-RDF11-CONCEPTS, 1.

3.1.20
triple
statement in the form *subject-predicate-object* (3.1.21, 3.1.22, 3.1.23) that expresses a relationship between two *resources* (3.1.14)

Note 1 to entry: The definition is adapted from W3C-RDF11-CONCEPTS, 3.1.

3.1.21
subject
resource (3.1.14) (an IRI) about which a statement is made in the form of an RDF *triple* (3.1.20)

Note 1 to entry: This term, as used in the ISO 21597 series, is part of the RDF(S)/OWL vocabulary, where each triple consists of a subject, a *predicate* (3.1.22) and an *object* (3.1.23); a set of such triples is called an RDF graph.

Note 2 to entry: The definition is adapted from W3C-RDF11-SCHEMA, 5.3.2.

3.1.22
predicate
the relationship between a *subject* (3.1.21) and an *object* (3.1.23) in an RDF *triple* (3.1.20), also called a property

Note 1 to entry: The definition is adapted from W3C-RDF11-SCHEMA, 5.3.3.

3.1.23
object
resource (3.1.14) (either an IRI or a literal) assigned as the specified property of the *subject* (3.1.21) in a *triple* (3.1.20)

Note 1 to entry: This term, as used in the ISO 21597 series, is part of the RDF(S)/OWL vocabulary, where each triple consists of a subject, a *predicate* (3.1.22) and an object; a set of such triples is called an RDF graph.

Note 2 to entry: The definition is adapted from W3C-RDF11-SCHEMA, 5.3.4.

3.2 Abbreviated Terms

DBF	DataBase File
GIS	Geographic Information System
GML	Geography Markup Language
GUID	Globally Unique Identifier
ICDD	Information Container for linked Document delivery
IFC	Industry Foundation Classes
IRI	Internationalized Resource Identifier
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SHACL	Shapes Constraint Language
SPARQL	Simple Protocol And RDF Query Language
SQL	Structured Query Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

NOTE IRI is an update of the URI released in 2005; while URIs are limited to a subset of the ASCII character set, IRIs can contain characters from the Universal Character Set (Unicode/ISO/IEC 10646). In the ISO 21597 series URIs and IRIs are used interchangeably.

4 Specifications

4.1 Use of RDF, RDFS and OWL constructs

All ontologies held in containers that conform to the ISO 21597 series shall be based on the languages RDF [W3C-RDF11-CONCEPTS], RDFS [W3C-RDF11-SCHEMA] and OWL [W3C-OWL2-SPEC] (referred to collectively in the ISO 21597 series as RDF(S)/OWL) and shall be serialized in RDF/XML [W3C-RDF11-XML] or any other equivalent RDF serialisation recommended by W3C.

It is expected that RDF(S)/OWL will be an important technology and a general platform for ontologies for the coming decades. Proprietary systems will increasingly adopt RDF(S)/OWL. However, to make the threshold for adoption of this document as low as possible, Annex C provides specifications to support the conversion of a container from RDF(S)/OWL to XSD/XML and vice versa.

In general, when used in the context of the world wide web, these languages use the following principles to support reasoning:

— Open world assumption - the truth of a statement is independent of whether it is known. In other words, not knowing that a statement is explicitly true does not imply that the statement is false.

— No unique names assumption - unless explicitly stated otherwise, it cannot be assumed that resources that are identified by different URIs are different.

The datasets that comply with the ontologies specified in the ISO 21597 series shall use the following interpretation of RDF(S)/OWL:

— Closed world assumption - a statement that is true is also known to be true; therefore, conversely, what is not formally specified in a container to be true, is false.

— Unique naming assumption - resources in a container that are identified with different URIs are considered to be different, unless explicitly declared as the same (using the *owl:sameAs* predicate).

Table 1 lists the RDF(S)/OWL constructs that are used in the ISO 21597 series and the interpretation to be used when validating the contents of a container. It is noted that, once the contents of the container has been validated, the data can be used in an open world context.

Table 1 — Listing of constructs used in the ISO 21597 series and their interpretation

Construct	Interpretation
owl:Class	In a dataset within a container, class membership for every individual shall be explicitly asserted, unless implicitly inferred using predicates such as *rdfs:subClassOf* [W3C-RDF11-SCHEMA, 3.4] or *owl:equivalentClass* [W3C-OWL2-SPEC, 9.1.2].
rdfs:subClassOf *rdfs:subPropertyOf*	The ISO 21597 series does not deviate from the W3C definitions [W3C-RDF11-SCHEMA]. Statements that may be inferred due to *rdfs:subClassOf* or *rdfs:subPropertyOf* statements shall be regarded as true even if not explicitly asserted. NOTE Statements where a class is mentioned are also true for any of its subclasses. Similarly, statements where a property is mentioned are also true for any of its sub properties.
owl:FunctionalProperty	The ISO 21597 series interprets *owl:FunctionalProperty* as a property with a maximum cardinality of 1. [W3C-OWL2-SPEC, 9.2.4]
owl:InverseFunctionalProperty	The ISO 21597 series interprets *owl:InverseFunctionalProperty* as an inverse property with a maximum cardinality of 1. [W3C-OWL2-SPEC, 9.2.7]
owl:equivalentClass	The ISO 21597 series does not deviate from the W3C definitions [W3C-OWL2-SPEC, 9.1.2]. Statements that may be inferred due to *owl:equivalentClass* statements shall be regarded as true even if not explicitly asserted.
rdfs:range *rdfs:domain*	These statements shall be interpreted as restrictions. It is invalid to have a subject or object of a statement (triple) in a dataset where that individual is a member of a class that does not comply with the *rdfs:range* or *rdfs:domain* declarations of the corresponding *owl:ObjectProperty* [W3C-OWL2-SPEC, 5.3] or *owl:DatatypeProperty* [W3C-OWL2-SPEC, 5.4]

Table 1 *(continued)*

Construct	Interpretation
owl:restriction *owl:onProperty* *owl:allValuesFrom* *owl:someValuesFrom* *owl:hasValue* *owl:cardinality* *owl:minCardinality* *owl:maxCardinality*	These statements shall be interpreted as restrictions. Any deviation from the specified restriction within a single container is considered invalid. NOTE As an example, if *owl:cardinality* is defined as 2, then a dataset that does not contain exactly 2 occurrences is not valid.
owl:inverseOf	The ISO 21597 series does not deviate from the W3C definitions [W3C-OWL2-SPEC, 9.2.4]. It is recommended that inverse properties are not asserted for individuals in a dataset. If they are asserted, they shall not contradict the assertions made in the opposite direction.
owl:disjointUnionOf	The expression shall be interpreted as a constraint where the subject is considered to be an abstract class in the sense that any individual member of the subject class shall also be a member of one (and only one) of the disjoint classes enumerated in the object part of the *owl:disjointUnionOf* statement. [W3C-OWL2-SPEC, 9.1.4]

4.2 Symbols and notations

Throughout the ISO 21597 series, the structure of the ontologies is illustrated using a UML notation. The purpose of this subclause is to describe that notation and the meaning of the terms and symbols that are used.

Tables 2 and 3 list the namespaces and corresponding prefixes used in the ISO 21597 series.

Table 2 — Namespaces and prefixes used in ontologies defined in the ISO 21597 series

Ontology	Prefix	Namespace
Container ontology	ct	https://standards.iso.org/iso/21597/-1/ed-1/en/Container
Linkset ontology	ls	https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset

Table 3 — Namespaces and prefixes used in ontologies referenced in the ISO 21597 series

Ontology	Prefix	Namespace
XML Schema	xsd	https://www.w3.org/2001/XMLSchema
Resource Description Framework	rdf	https://www.w3.org/1999/02/22-rdf-syntax-ns
RDF Schema	rdfs	https://www.w3.org/2000/01/rdf-schema
Web Ontology Language	owl	https://www.w3.org/2002/07/owl

Figure 1 illustrates the UML notations used in the ISO 21597 series to render classes and properties.

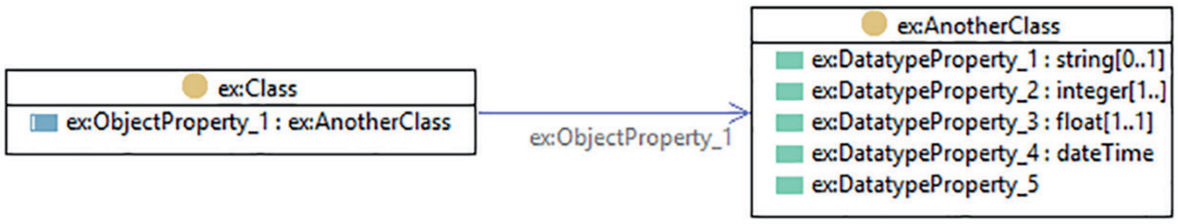

ex:Class
ex:ObjectProperty_1 : ex:AnotherClass
ex:ObjectProperty_1
ex:AnotherClass
ex:DatatypeProperty_1 : string[0..1]
ex:DatatypeProperty_2 : integer[1..]
ex:DatatypeProperty_3 : float[1..1]
ex:DatatypeProperty_4 : dateTime
ex:DatatypeProperty_5

Figure 1 — UML notation for classes and properties

A class (*owl:Class*) is illustrated by a rectangular box with two compartments as shown in Figure 1. In the upper compartment, the class name ("*ex:Class*" in Figure 1) is displayed. Note that the class name is shown following the pattern "*prefix:ClassName*", where the prefix ("ex" in the example) denotes namespace of the ontology and "ClassName" is the name of the class. The prefixes actually used in the ISO 21597 series are defined in Tables 2 and 3.

The lower compartment shows the specified properties for that class. There are two general types of properties:

— Datatype properties are those for which the value is a data literal, as illustrated for *ex:AnotherClass* in Figure 1; and

— Object properties, for which the value is an individual; e.g. *ex:Class* in Figure 1, where the property *ex:ObjectProperty_1* references an individual of class *ex:AnotherClass*.

The property definitions are shown according to the pattern "*prefix:propertyName: range[cardinality]*". The range of a datatype property shall be based on one of the predefined data types in XML schema [W3C-XML-DATATYPES]. The range of an object property is usually one of the classes occurring in the ontology but may also refer to a class in another ontology.

If classes on both sides (domain and range) of an object property are visible in a diagram, the object property may also be illustrated with an (blue) arrow between the classes pointing from the domain class towards the range class (as shown in Figure 1). The name of the object property is displayed along the arrow as well as in the property compartment of the class box as explained above.

Any cardinality restrictions are displayed within square brackets using the following notation: *[minCardinality..maxCardinality]*, where *minCardinality* specifies the minimum allowed occurrences and *maxCardinality* specifies the maximum allowed number of occurrences

The cardinality restrictions shall be interpreted in the following fashion:

— omitted - no cardinality restriction exists, i.e. any number of occurrences from zero to many are allowed;

— *maxCardinality* omitted (e.g. [0..], [1..] etc) - maximum cardinality is unrestricted.

If two classes are related using an *rdfs:subClassOf* predicate, this is rendered using an arrow as shown in Figure 2. This diagram illustrates that *ex:SubClass* and *ex:Class* are related using an *rdfs:subClassOf* predicate.

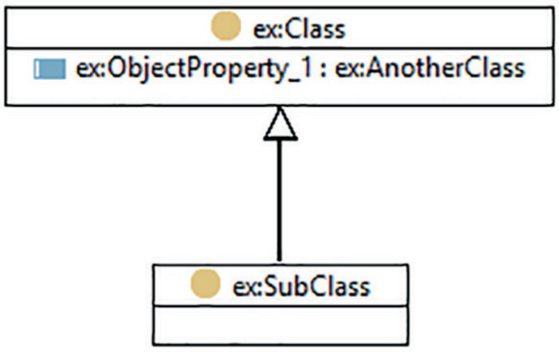


Figure 2 — Depiction of a sub-class relationship

Disjoint classes are illustrated in Figure 3.

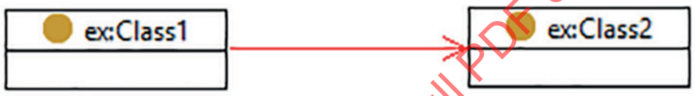


Figure 3 — Depiction of disjoint classes

The red arrow pointing from *ex:Class1* to *ex:Class2* declares that they are disjoint, meaning that an instance is not allowed to be a member of both *ex:Class1* and *ex:Class2*. This is declared with an *owl:disjointUnionOf* statement (*ex:Class1 owl:disjointUnionOf ex:Class2*). The *owl:disjointUnionOf* property is symmetric, meaning that if Class1 is disjoint with Class2, then Class2 is also disjoint with Class1.

Two classes which are declared as equivalent by the use of *owl:equivalentClass* (e.g. *ex:Class3 owl:equivalentClass ex:Class4*) are depicted as shown in Figure 4.

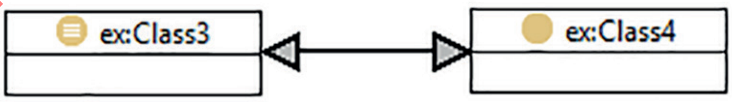


Figure 4 — Depiction of equivalent classes

Finally, a class may be both the *Domain* and *Range* for a certain *ObjectProperty*. Such a relationship is rendered as shown in Figure 5, i.e. with the little arrow aligned to the bottom of the class box without any label attached to the arrow.

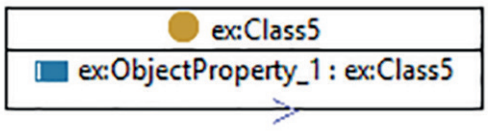


Figure 5 — Depiction of an ObjectProperty defined by an individual of the same class

4.3 Container structure

4.3.1 Overview

A container is a file that shall have an extension ".icdd" and shall comply with ISO/IEC 21320-1 also known as ZIP64.

A container includes a header file in the top-level folder; this file shall comply with the RDF(S)/OWL standards and shall be serialized in RDF/XML [W3C-RDF11-XML] or any other equivalent RDF serialisation recommended by W3C. The name of this header file is Index.rdf.

As a minimum, a container shall have at least three folders as illustrated in Figure 6. The purpose of these top-level folders is explained in the following subclauses. The "Payload documents" folder and "Payload triples" folder may contain nested folders to allow groups of associated digital resources to be held together and referenced as a group (e.g. a building information model with its associated reference files or a set of linked spreadsheets).

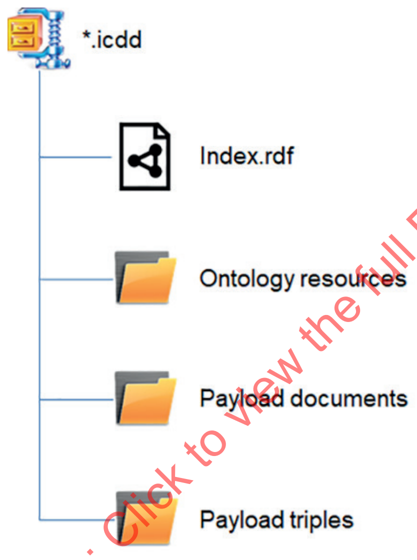


Figure 6 — Minimum structure of the root of a container

Figure 7 shows the hierarchy of folders and files within a container.

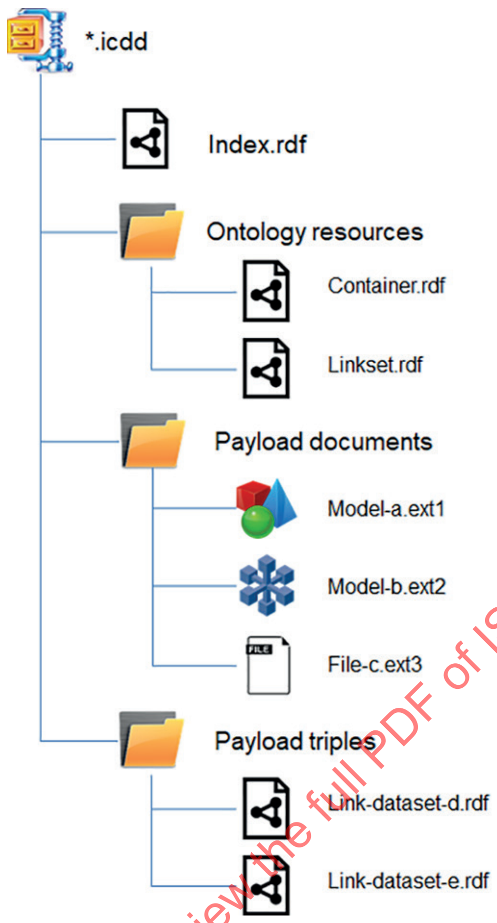


Figure 7 — Hierarchy of folders and files in a container

4.3.2 “Ontology resources” folder

The “Ontology resources” folder can be used to store the Linkset.rdf and Container.rdf ontologies that together provide the object classes and properties that shall be used to specify the contents of and links between the documents within the container. These ontologies shall be serialized in the RDF/XML format [W3C-RDF11-XML] or any other equivalent RDF serialisation recommended by W3C. Since they are both available as downloadable files (and therefore can be referenced as resources held locally but external to the container), it is not mandatory to include them here, but if included, the files in this folder shall take precedence.

4.3.3 “Payload documents” folder

The “Payload documents” folder shall be used for storing all the documents that are included in the container (referred to as the payload). Sub-folders are allowed.

4.3.4 “Payload triples” folder

The “Payload triples” folder shall be used for storing Linkset files and may include sub-folders.

4.4 Ontologies and datasets

4.4.1 Overview

Using RDF(S)/OWL technology, the object classes and properties that are used to specify the contents of and links between the documents within the container is specified within the container using two ontologies:

— Container ontology, available online via https://standards.iso.org/iso/21597/-1/ed-1/en/Container.rdf.

— Linkset ontology, available online via https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset.rdf.

As noted previously, these ontologies can be included in the container in the “Ontology resources” folder to create a self-contained container that may be used off-line or for archiving purposes. Since these ontologies conform to the ISO 21597 series, they shall not be modified in any way.

In addition, the container shall include RDF(S)/OWL datasets that describe the contents of the container and the links between those documents. We distinguish two types of datasets. First, the container shall include one Index dataset, called Index.rdf and held in the root of the container; it is used to describe the container and to specify the documents that make up its contents. Second, the container may include zero or more Link datasets, used to specify the link relationships among documents. All such datasets shall reference the Container and Linkset ontologies in Annex E.

The Container and Linkset ontologies are described in 4.4.2 and 4.4.3. The two types of dataset described in the previous paragraph are then described in 4.4.4 and 4.4.5 respectively.

4.4.2 Container ontology

The Container ontology is an RDF(S)/OWL file containing definitions of the classes and properties used in an Index dataset, providing metadata about the container.

The Index dataset enables the specification of:

— version of ICDD standard via the import of the reference ontology;

— list of external documents with metadata:

  — mandatory file name (including a URI or IRI);

  — optional format;

  — optional description;

— list of internal documents with metadata:

  — mandatory file name (including its path in the container folder structure);

  — optional format;

  — optional description;

— reference to Link datasets.

Tables 4, 5 and 6 list the objects, datatype properties and object properties respectively that are used in the Container ontology, providing brief descriptions of each.

Table 4 — Classes defined in the Container ontology

Object name	Description
ct:ContainerDescription	A description for a container where all documents are listed and where Link datasets can be found. There shall be exactly one *ct:ContainerDescription* instance in any container.
ct:EncryptedDocument	A reference to an encrypted document.
ct:ExternalDocument	A reference to a document outside a container.
ct:InternalDocument	A reference to a document inside a container.
ct:Linkset	A reference to an RDF(S)/OWL file containing links.
ct:Document	An abstract class for references to a document; an individual shall be a member of *ct:ExternalDocument* or *ct:InternalDocument*; and optionally, individuals may also be a member of other subtypes of *ct:Document* such as *ct:SecuredDocument* and/or *ct:EncryptedDocument.*
ct:SecuredDocument	A document secured by a checksum algorithm (see also properties *ct:checksum* and *ct:checksumAlgorithm*).
ct:FolderDocument	A document comprising multiple files located in one folder, such as a GIS dataset consisting SHP files with associated DBF files.
ct:Party	An abstract class that represents the generalization of a *ct:Organisation* or a *ct:Person*; entities can refer to an individual of a subclass of *ct:Party* via the *ct:creator*, *ct:modifier* or *ct:publisher* object properties.
ct:Person	A class representing a person for provenance purposes.
ct:Organisation	A class representing an organization for provenance purposes.

Table 5 — Datatype properties used in the Container ontology

Datatype name	Description
ct:checksum	A checksum hash for the document reference; the checksum algorithm is specified by the property *ct:checksumAlgorithm*.
ct:checksumAlgorithm	The algorithm used to generate the checksum hash.
ct:conformanceIndicator	A string-based indicator for *ct:ContainerDescription* to show to which part of the ISO 21597 series this container conforms: for a Part 1 container, the value should be set to "ICDD-Part1-Container"; the range is not restricted to allow other indicator values.
ct:creationDate	The creation date as *xsd:dateTime*.
ct:description	A general description.
ct:encryptionAlgorithm	Optional string describing the encryption.
ct:filename	The file name of a *ct:Linkset* or *ct:InternalDocument*; the root corresponds with the “Payload documents” folder of the ICDD container; the forward slash character ("/") shall be used as a folder separator. NOTE An example of a *ct:filename* is “IFC Models/MyFile_1.ifc” which refers to the file MyFile_1.ifc inside the folder IFC Models inside the “Payload documents” folder in the container.
ct:foldername	A folder name for specifying a folder where a multi file document can be found; the root corresponds with the “Payload documents” folder of the ICDD container; the forward slash character ("/") shall be used as a folder separator. NOTE An example of a *ct:foldername* is “GIS Datasets/Terrain” which refers to the folder Terrain inside folder GIS Datasets inside the “Payload documents” folder in the container.
ct:filetype	A string that specifies the file type such as "GML", "IFC", "shp", "xlsx", "pdf","rvt"; the string may be a compound string to indicate version and data format (e.g. “ifc-4-xml-zip”).

Table 5 *(continued)*

Datatype name	Description
ct:format	The media-type of a document shall follow the specification by the Internet Assigned Numbers Authority [IANA]; examples are “application/pdf” and “audio/mpeg”.
ct:modificationDate	The modification date as *xsd:dateTime*.
ct:name	A name for a document. NOTE An example of a *ct:name* is “D101”.
ct:requested	A boolean to indicate whether a document is required or not; when this property is not set the value can be interpreted as “false”.
ct:url	The URL where the external document can be found.
ct:userID	The user defined identifier.
ct:versionDescription	An optional character string that may be used to provide a description for a version of the corresponding resource.
ct:versionID	An optional character string that may be used to identify a version of the corresponding resource.

Table 6 — Object properties used in the Container ontology

Object property name	Description
ct:createdBy	A reference to a creator of this instance which can only be a subclass of *ct:Party*. Inverse property: *ct:created*.
ct:modifiedBy	A reference to the modifier of this instance which can only be a subclass of *ct:Party*. Inverse property: *ct:modified*.
ct:publishedBy	The party responsible for making the container available. Inverse property: *ct:published*.
ct:alternativeDocument	A property to link a document to an alternative version of that document. Inverse property: *ct:alternativeTo*.
ct:belongsToContainer	An owl property defining the relation between a document reference and a container.
ct:containsLinkset	A relation from a *ct:ContainerDescription* to a *ct:Linkset* reference. Multiple linkset references are allowed. Inverse property: *ct:containedInContainer*.
ct:containsDocument	A relation from *ct:ContainerDescription* to a document reference. Relations to multiple document references are allowed.
ct:priorVersion	An optional reference to the prior version of this resource. Inverse property: *ct:nextVersion*.

Figure 8 illustrates the context of the Container ontology, showing the structure of a container and the meta-information associated with it. The *ct:ContainerDescription* class is a subclass of the *owl:Ontology* class (more information on *owl:Ontology* can be found in section 3 of the OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax [W3C-OWL2-SPEC]). Each ontology is an individual that is a member of the *owl:Ontology*.

The Index dataset of a container declares an individual that is a type of *ct:ContainerDescription* and thus also a member of the class *owl:Ontology*. The IRI or URI of this individual is used as the identifier for the Index dataset and consequently can be used to identify the container. The Index dataset imports the Container ontology via *owl:imports* property, so that the individual inherits a *ct:description* property and mandatory *ct:publisher* object property referring to a *ct:Party*. It also refers to individuals of classes *ct:Linkset* and *ct:Document* via the object properties *ct:containsLinkset* and *ct:containsDocument* respectively, in that way defining the structure of the container.

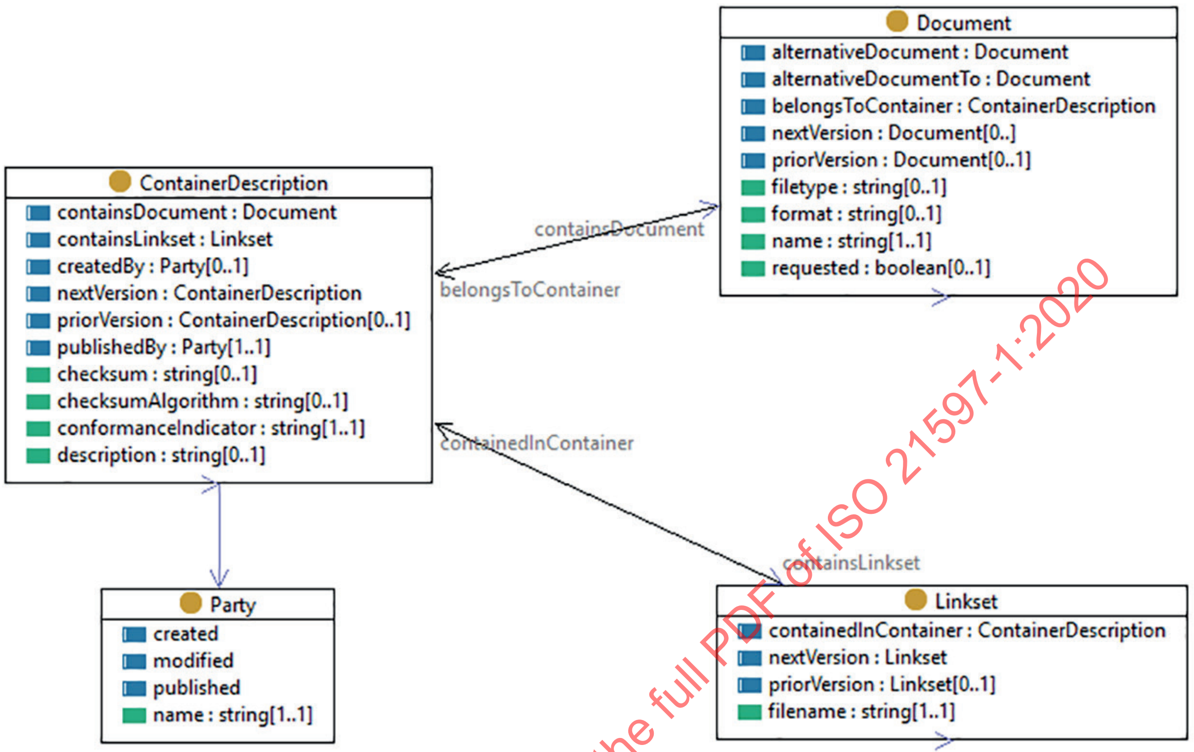

Document
alternativeDocument : Document
alternativeDocumentTo : Document
belongsToContainer : ContainerDescription
nextVersion : Document[0..]
priorVersion : Document[0..1]
filetype : string[0..1]
format : string[0..1]
name : string[1..1]
requested : boolean[0..1]
ContainerDescription
containsDocument : Document
containsLinkset : Linkset
createdBy : Party[0..1]
nextVersion : ContainerDescription
priorVersion : ContainerDescription[0..1]
publishedBy : Party[1..1]
checksum : string[0..1]
checksumAlgorithm : string[0..1]
conformanceIndicator : string[1..1]
description : string[0..1]
containsDocument
belongsToContainer
containedInContainer
containsLinkset
Party
created
modified
published
name : string[1..1]
Linkset
containedInContainer : ContainerDescription
nextVersion : Linkset
priorVersion : Linkset[0..1]
filename : string[1..1]

Figure 8 — *ct:ContainerDescription* context

Figure 9 illustrates the properties and sub-classes of documents supported in a container that conforms to the ISO 21597 series. All documents shall have a *ct:name* and optionally *ct:filetype*, *ct:format* and *ct: requested* properties (the latter being a boolean to indicate that the document is requested from the recipient of the container). Individuals of class *ct:Document* shall be typed as either a *ct:InternalDocument* or *ct:ExternalDocument* (enforced by an *owl:disjointUnionOf* statement interpreted as a restriction). The figure shows additional subclasses of *ct:Document* that may be used to declare encrypted, secure or folder documents.

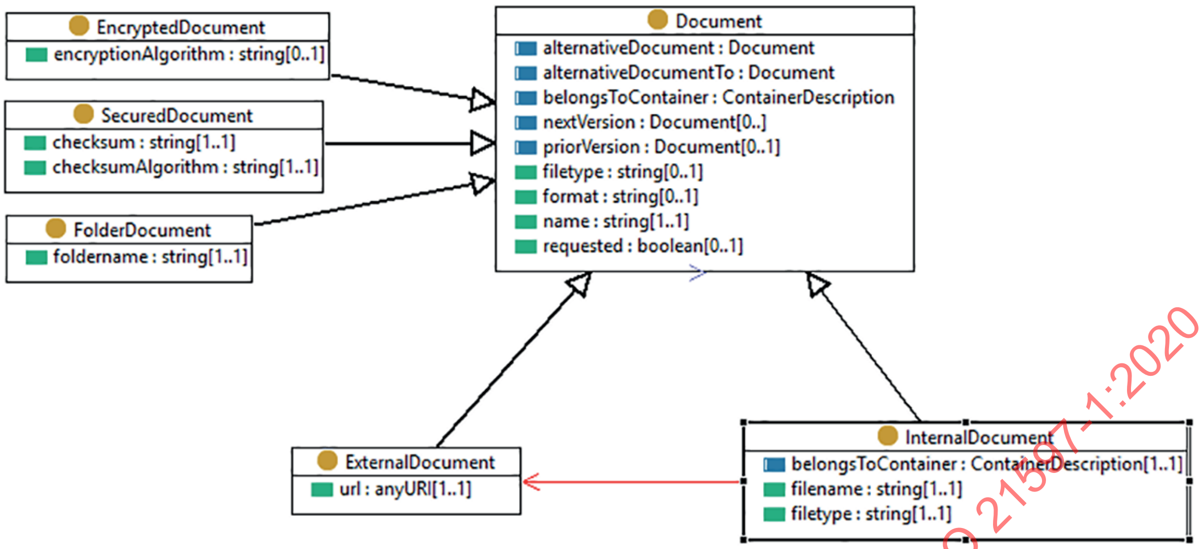

EncryptedDocument
encryptionAlgorithm : string[0..1]
SecuredDocument
checksum : string[1..1]
checksumAlgorithm : string[1..1]
FolderDocument
foldername : string[1..1]
Document
alternativeDocument : Document
alternativeDocumentTo : Document
belongsToContainer : ContainerDescription
nextVersion : Document[0..]
priorVersion : Document[0..1]
filetype : string[0..1]
format : string[0..1]
name : string[1..1]
requested : boolean[0..1]
ExternalDocument
url : anyURI[1..1]
InternalDocument
belongsToContainer : ContainerDescription[1..1]
filename : string[1..1]
filetype : string[1..1]

Figure 9 — *ct:Document* context

Subject to which subtypes are specified, new properties are available or are mandatory. Multiple typing of documents is allowed unless an *owl:disjoint* statement between these classes is present. A *ct:FolderDocument* represents a document that is present in a folder and may comprise multiple files. A *ct:EncryptedDocument* class is used to state that a document is encrypted. A *ct:SecuredDocument* can be used when a document is secured by a hash using an algorithm such as for example SHA256.

4.4.3 Linkset ontology

The Linkset ontology is an RDF(S)/OWL file that provides the object classes and properties that shall be used to create a Link dataset. A Link dataset specifies the linkages among documents and re-uses the document descriptions from the Index dataset, and therefore shall import the Index dataset.

A Link can specify interdependencies among two or more documents as well as among elements contained in these documents. Most typical applications of Links are linkages:

— between a single element and a related document (e.g. a row in a spreadsheet and a GML file);

— between one element in one document and multiple related elements in other documents (e.g. a catalogue element and the occurrences in a building model);

— among a set of elements in one document and related elements in multiple documents (e.g. a group columns and the corresponding specification items in a bill of quantity and the processes in schedule document).

To identify certain elements within a document, this part of the ISO 21597 series provides three general mechanisms: a string-based identifier; a query; or a URL-based identifier. The choice of the element attributes used and syntax of identifiers and queries is left up to implementers.

Tables 7, 8 and 9 list the objects, datatype properties and object properties respectively that are used in the Linkset ontology, providing brief descriptions of each.

Table 7 — Classes defined in the Linkset ontology

Object name	Description
ls:BinaryLink	An *ls:Link* comprising exactly 2 individuals of class *ls:LinkElement*.
ls:DirectedLink	An *ls:Link* that uses the subproperties *ls:hasFromLinkElement* and *ls:hasToLinkElement* to denote a direction of this link.
ls:DirectedBinaryLink	A subtype of a binary link (that has exactly 2 instances of *ls:LinkElement*) that uses the subproperties *ls:hasFromLinkElement* and *ls:hasToLinkElement* to denote a direction of this link.
ls:Directed1toNLink	A subtype of *ls:DirectedLink* mandating exactly 1 *ls:hasFromLinkElement*.
ls:Identifier	An abstract class for identifying an element within a document; in cases where an identifier may be computed, this shall be managed by the implementer since no method is specified in the ISO 21597 series. *ls:Identifier* is the union of its disjoint subclasses *ls:StringBasedIdentifier*, *ls:URIBasedIdentifier* and *ls:QueryBasedIdentifier*. Considering the *owl:disjointUnionOf* constraint definition (see Table 1), any individual member of this class shall be a member of one and only one of its subclasses.
ls:Link	A grouping of 1 or more instances of *ls:LinkElement*.
ls:LinkElement	A class for referencing to a document or to an element in a document.
ls:QueryBasedIdentifier	An identifier of an element in a document based upon a query.
ls:StringBasedIdentifier	Identification of an element within a document via a String ID.
ls:URIBasedIdentifier	URI/IRI-based identifier for a document, or element within a document, that is located on the web.

Table 8 — Datatype properties defined in the Linkset ontology

Datatype name	Description
ls:identifier	A datatype String property containing the actual ID string.
ls:identifierField	A String datatype for defining the field(s) where the identifier can be found; in cases where the identifier is composed of multiple fields, the implementer shall choose the syntax rules.
ls:queryExpression	The query resulting in an identifier.
ls:queryLanguage	A query language specification.
ls:uri	A URI/IRI for referring to a document.

Table 9 — Object properties defined in the Linkset ontology

Objecttype name	Description
ls:hasIdentifier	A relation from *ls:LinkElement* to an *ls:Identifier*.
ls:hasLinkElement	A relation from an *ls:Link* to an *ls:LinkElement*.
ls:hasFromLinkElement	A relation from an *ls:Link* to an *ls:LinkElement*. It is a sub property of *ls:hasLinkElement*.
ls:hasToLinkElement	A relation to an *ls:Link* from an *ls:LinkElement*. It is a sub property of *ls:hasLinkElement*.
ls:hasDocument	A relation from a *ls:LinkElement* to a *ct:Document*.

Figure 10 shows the basic structure for an *ls:Link* dataset.

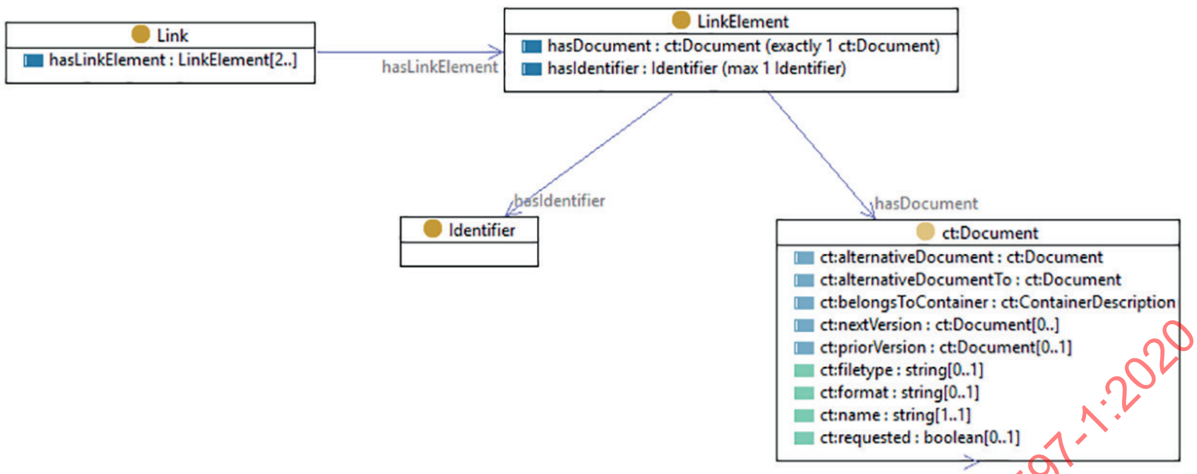

Link
hasLinkElement : LinkElement[2..]
hasLinkElement
LinkElement
hasDocument : ct:Document (exactly 1 ct:Document)
hasIdentifier : Identifier (max 1 Identifier)
hasIdentifier
hasDocument
Identifier
ct:Document
ct:alternativeDocument : ct:Document
ct:alternativeDocumentTo : ct:Document
ct:belongsToContainer : ct:ContainerDescription
ct:nextVersion : ct:Document[0..]
ct:priorVersion : ct:Document[0..1]
ct:filetype : string[0..1]
ct:format : string[0..1]
ct:name : string[1..1]
ct:requested : boolean[0..1]

Figure 10 — *ls:Link* context

An *ls:Link* has at least 2 instances of *ls:LinkElement* (*ls:hasLinkElement*). An *ls:LinkElement* references a *ct:Document* (*ls:hasDocument*) and optionally an identifier (*ls:hasIdentifier*). The identifier reference is necessary for deep linking, i.e. referencing individual elements within a *ct:Document* and is explained further in Figure 11.

ls:Link has subclasses *ls:BinaryLink* and *ls:DirectedLink*.

Figure 11 shows the structure for *ls:BinaryLink*, with exactly two instances of *ls:LinkElement* via the *ls:hasLinkElement* property. Furthermore, an *ls:DirectedBinaryLink* is defined as a subclass of *ls:BinaryLink* with mandatory properties *ls:hasFromLinkElement* and *ls:hasToLinkElement*. These properties are subproperties of *ls:hasLinkElement*. An *ls:DirectedBinaryLink* is a subclass of both *ls:BinaryLink* and *ls:DirectedLink*, inheriting all restrictions.

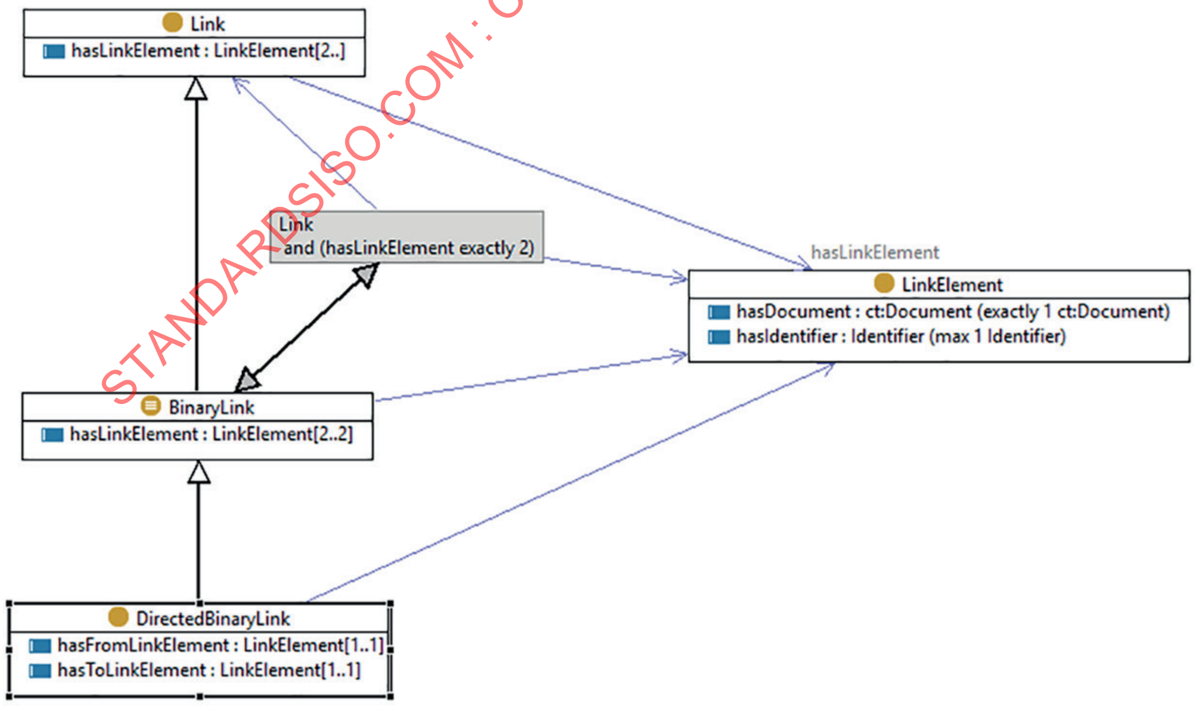

Link
hasLinkElement : LinkElement[2..]
Link
and (hasLinkElement exactly 2)
hasLinkElement
LinkElement
hasDocument : ct:Document (exactly 1 ct:Document)
hasIdentifier : Identifier (max 1 Identifier)
BinaryLink
hasLinkElement : LinkElement[2..2]
DirectedBinaryLink
hasFromLinkElement : LinkElement[1..1]
hasToLinkElement : LinkElement[1..1]

Figure 11 — *ls:BinaryLink* context

Figure 12 shows the structure for *ls:DirectedLink*, i.e. a relationship where the direction of the relationship is significant. Here an *ls:Directed1toNLink* is defined as a subclass of *ls:DirectedLink* with an additional “exactly 1” restriction on *ls:hasFromLinkElement* property.

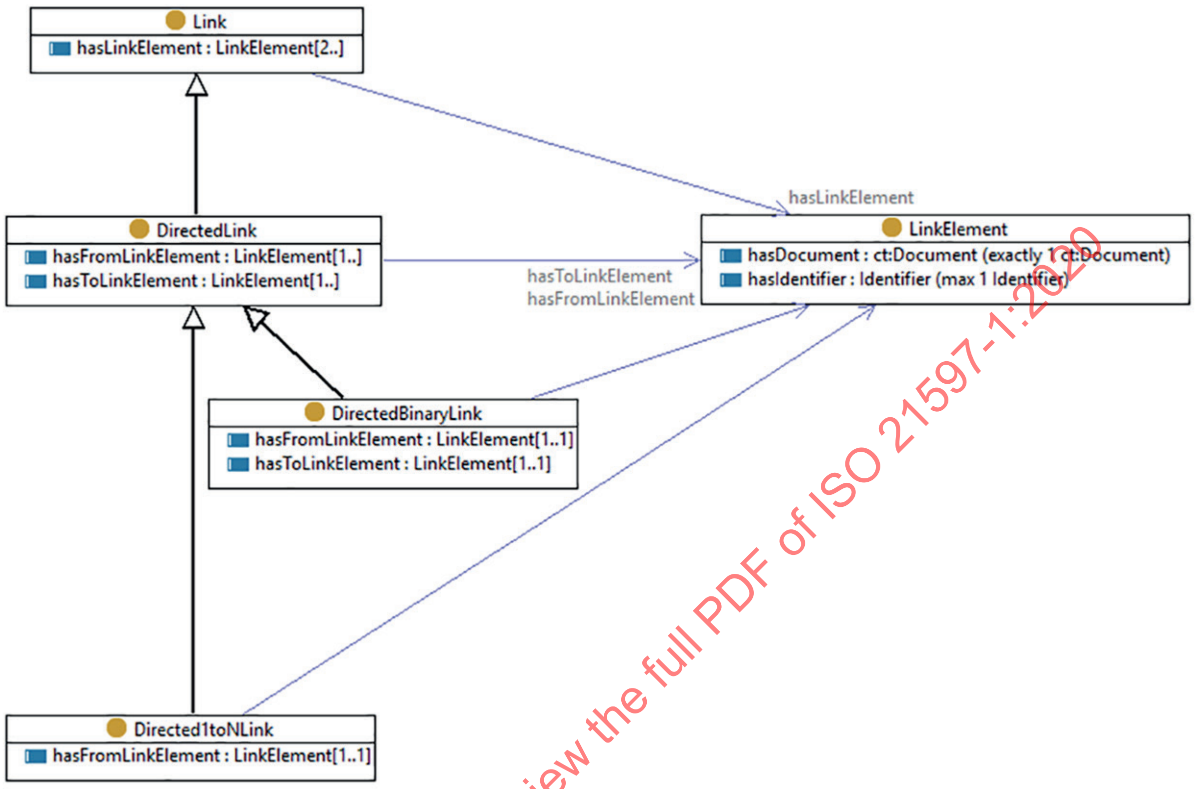

Link
hasLinkElement : LinkElement[2..]
hasLinkElement
DirectedLink
hasFromLinkElement : LinkElement[1..]
hasToLinkElement : LinkElement[1..]
hasToLinkElement
hasFromLinkElement
LinkElement
hasDocument : ct:Document (exactly 1 ct:Document)
hasIdentifier : Identifier (max 1 Identifier)
DirectedBinaryLink
hasFromLinkElement : LinkElement[1..1]
hasToLinkElement : LinkElement[1..1]
Directed1toNLink
hasFromLinkElement : LinkElement[1..1]

Figure 12 — *ls:DirectedLink* context

Figure 13 shows the objects and properties for *ls:Identifier*.

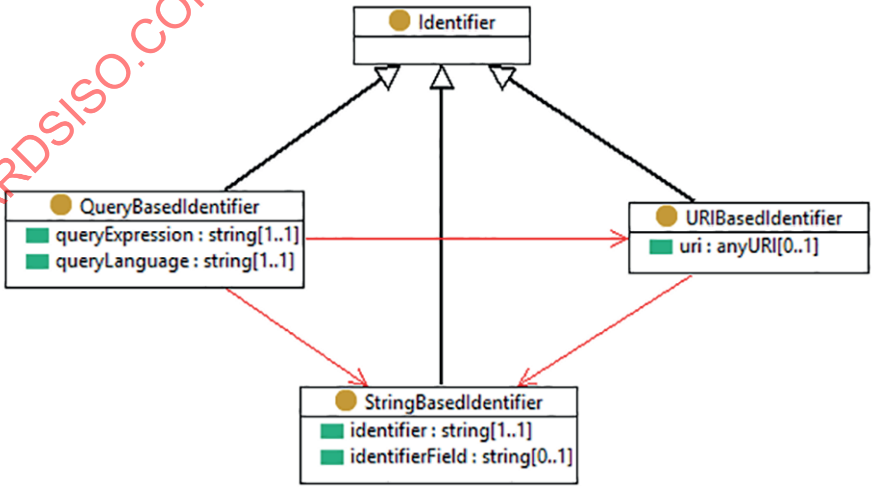

Identifier
QueryBasedIdentifier
queryExpression : string[1..1]
queryLanguage : string[1..1]
URIBasedIdentifier
uri : anyURI[0..1]
StringBasedIdentifier
identifier : string[1..1]
identifierField : string[0..1]

Figure 13 — *ls:Identifier* context

ls:Identifier is the union of its disjoint subclasses *ls:StringBasedIdentifier*, *ls:URIBasedIdentifier* and *ls:QueryBasedIdentifier*. In the ISO 21597 series, this is interpreted as a constraint, having the effect of making *ls:Identifier* an abstract superclass that shall be realized through one of those three subclasses.

The *ls:StringBasedIdentifier* has a mandatory *ls:Identifier* data type (*xsd:string*) for the actual ID. The optional *ls:identifierField* can be used to supply extra information about the field, such as where to find the ID.

The *ls:URIBasedIdentifier* can be used to refer to a document or an element within a document using an *xsd:anyUri* via the *ls:uri* datatype property.

The *ls:QueryBasedIdentifier* has a mandatory *ls:queryExpression* datatype property, a mandatory *ls:queryLanguage* datatype property and an optional *ls:querySortExpression*. An SQL query (such as "Select ID from table where Country='Mexico'") could be the input for the *ls:queryExpression* and "SQL" could be the query language. Alternatively an XQuery (such as "for $x in doc("costestimation.xml")/foundation/objects where $x/price>30 order by $x/id return $x/id") could be the input the *ls:queryExpression*, "Xquery" could be the query language.

4.4.4 Index dataset

A container shall have one Index dataset called Index.rdf. The Index dataset shall reside in the root of the container.

The Index dataset shall import the Container ontology via *owl:imports* statement in the *owl:Ontology* resource. The *owl:Ontology* individual within this graph will be typed as a *ct:ContainerDescription* and its properties can be filled in. Individuals of *ct:Document* can be attached to this description according to the Container ontology. Each individual of a *ct:Document* describes a document. In case of an internal document a reference to its location in the "Payload documents" folder is mandatory.

4.4.5 Link dataset

Any Link dataset(s) included in the container shall reside in the "Payload triples" folder and shall import at least Index.rdf and the Linkset ontology. It contains all the links between documents as individuals of *ls:Link* and *ls:LinkElement* according to the Linkset ontology.

4.5 Versioning

The purpose of versioning within the ISO 21597 series is to enable the following functionalities:

— enable the delivery of various alternative solutions in one container (e.g. when proposing different design scenarios);

— enable transfer of the history of resources;

— enable exchange of information about versions between partners to provide a means to reference particular states of the information;

— enabling tracing of previous versions.

Versioning is handled by adding versioning properties to the ontologies and restricting them to certain domains and ranges.

The following subjects may be versioned:

— an individual which is a member of class *ct:ContainerDescription* as specified in the Container ontology; version information at this level, if provided, specifies a version for the container as a whole;

— an individual which is a member of class *ct:Document* as specified in the Container ontology; version information at this level, if provided, specifies a version for the referenced document;

— an individual which is a member of *ct:Linkset* as specified in the Container ontology; version information at this level specifies a version for a particular linkset.

If a resource is versioned, the property *ct:versionID* shall be used to indicate the version of the resource and the corresponding property *ct:versionDescription* may be used to include a clarifying description of the version. This property is declared as functional and only one value is allowed. The *ct:versionID* property shall have the range *xsd:string*. The ISO 21597 series does not prescribe any particular formatting of the character string for the content of version identification, leaving that up to the user.

To enable tracking of version history for resources, a versioned resource may, besides *ct:versionID*, indicate a prior version using the property *ct:priorVersion*. The object of a *ct:priorVersion* predicate shall have the same type as its subject.

NOTE Versioning of data elements inside particular documents of particular formats (e.g. ENTITY instances in .ifc-files) is out of scope for the ISO 21597 series.

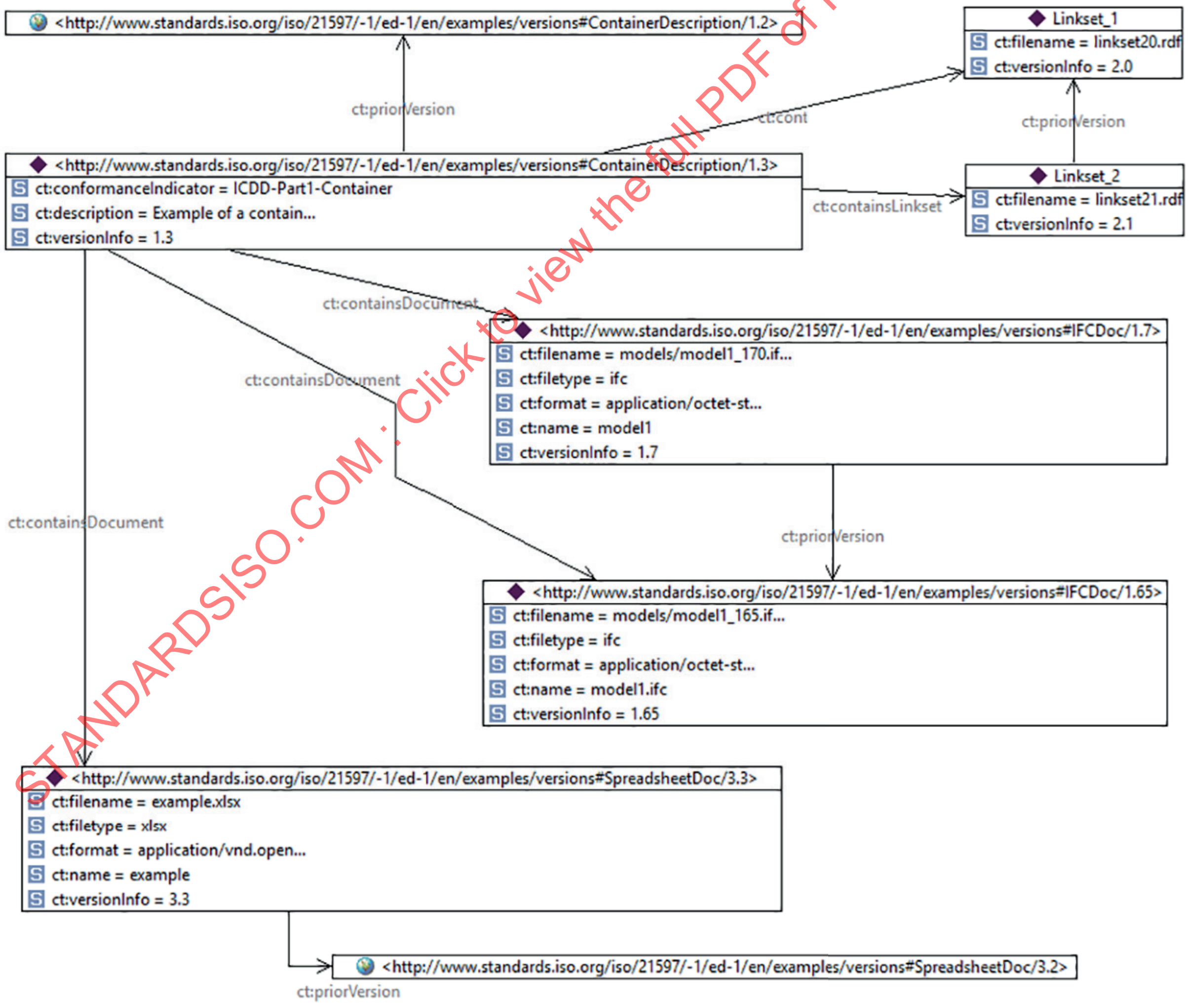
Figure 14 provides an example dataset with version information.

— an Index dataset, version 1.3, which references a prior version of the same container;

— the container references a number of Link datasets and individuals of class *ct:InternalDocument*;

— two Link datasets, where one Link dataset is the *ct:priorVersion* of the other Link dataset;

— three Documents: two IFC documents where one is the *ct:priorVersion* of the other; one spreadsheet document referencing its *ct:priorVersion* (not present in this container).



<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#ContainerDescription/1.2>
ct:priorVersion
ct:cont
Linkset_1
ct:filename = linkset20.rdf
ct:versionInfo = 2.0
ct:priorVersion
<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#ContainerDescription/1.3>
ct:conformanceIndicator = ICDD-Part1-Container
ct:description = Example of a contain...
ct:versionInfo = 1.3
ct:containsLinkset
Linkset_2
ct:filename = linkset21.rdf
ct:versionInfo = 2.1
ct:containsDocument
<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#IFCDoc/1.7>
ct:filename = models/model1_170.if...
ct:filetype = ifc
ct:format = application/octet-st...
ct:name = model1
ct:versionInfo = 1.7
ct:containsDocument
ct:containsDocument
ct:priorVersion
<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#IFCDoc/1.65>
ct:filename = models/model1_165.if...
ct:filetype = ifc
ct:format = application/octet-st...
ct:name = model1.ifc
ct:versionInfo = 1.65
<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#SpreadsheetDoc/3.3>
ct:filename = example.xlsx
ct:filetype = xlsx
ct:format = application/vnd.open...
ct:name = example
ct:versionInfo = 3.3
<http://www.standards.iso.org/iso/21597/-1/ed-1/en/examples/versions#SpreadsheetDoc/3.2>
ct:priorVersion

Figure 14 — Example data with version information

4.6 Additional properties in datasets

Individuals in a dataset shall only be members of classes defined in the ISO 21597 series. However, it is permitted to use other predicates defined outside of the Container or Linkset namespaces to define additional properties. These predicates and their objects can be used to exchange additional information for individuals in the datasets.

The object can be a literal or a URI. This document prescribes no further interpretation of these additional properties and consequently each shall be handled as a key-value pair. Use case 1B in Annex A includes an example of an additional property.

5 Conformance requirements

Any container file which is claimed to be in conformance with this document shall satisfy all requirements in this clause.

The following requirements apply for the container:

1. The container shall be a valid ZIP64 file.
2. The container shall have an Index.rdf file in the root.
3. The container shall have the following folders:
    a. "Ontology resources";
    b. "Payload documents";
    c. "Payload triples".
4. If the ontology file Container.rdf and/or Linkset.rdf is included in the container, then it shall be located in the "Ontology resources" folder.
5. The container shall have the ".icdd" filename extension.

The following requirements apply for the Index.rdf file:

1. It shall comply with RDF(S)/OWL.
2. It shall be serialized in RDF/XML [W3C-RDF11-XML] or any other equivalent RDF serialisation recommended by W3C.
3. It shall comply with the Container.rdf file as specified in this part of the ISO 21597 series.
4. It shall import the Container.rdf ontology using *owl:imports* predicate.
5. It shall list all internal documents and external documents.
6. It shall list all datasets.
7. It shall contain the value "ICDD-Part1-Container" for the *ct:conformanceIndicator* property.

The following requirements apply for every Link dataset file:

1. It shall comply with RDF(S)/OWL.
2. It shall be serialized in RDF/XML [W3C-RDF11-XML] or any other equivalent RDF serialisation recommended by W3C.
3. It shall comply with the Linkset.rdf ontology as specified in this part of the ISO 21597 series.
4. It shall be stored in the "Payload triples" folder.

The following requirement applies for every document:

Each document contained in the container, shall be stored in the “Payload documents” folder.

The following requirement applies to the extendability of the ontologies:

No extensions are permitted, therefore the Index.rdf file and any Link dataset file may only contain individuals that are in compliance with the classes as specified by Container.rdf and Linkset.rdf.

NOTE For validating RDF graphs, W3C offers a Shapes Constraint Language (SHACL) which is an official W3C recommendation since 20 July 2017. It is envisaged that in the future SHACL will be of great importance for the formal validation of RDF files. Since it is a new technology, this document only includes an example in Annex D.

Annex A
(informative)

Use cases

A.1 Context

The use cases concern the business process of a public client in the infrastructure sector. This client manages the infrastructure in a particular region. As part of the management process, periodic inspections are performed. The procedure for inspections is described in a manual. Inspections are outsourced to contractors. The use cases refer to the process on the interface between client and contractor. The client is using an ICDD container to make available the exchange requirements to the contractor; the contractor is using an ICDD container to deliver the required information.

A.2 Use case 1A — Delivery of documents

A.2.1 General

The client requests the contractor to perform a maintenance inspection of a viaduct (shown in Figure A.1) known as asset with identification code 48D-100. Upon completion of the assignment, the client asks for an information delivery with the following content:

— inspection report (Excel, one line per construction part);

— one or more photos for every issue reported;

— a 3D model (to visualize the construction - in IFC format);

— a timesheet (to report hours worked on this assignment - in Excel format).

For the purpose of the assignment, the client provides/sends a container, equipped with slots for the requested documents. The information delivery specification is included in the container, as well as a template for the inspection report and the timesheet.

The decomposition of a viaduct is required to be as follows:

— viaduct:
  — foundation;
  — load-bearing structure:
    — main girders;
    — bridge deck;
  — support:
    — abutments;
    — pillars;
  — expansion joint.

The template for an inspection report specifies the following data elements:

— name of the construction;

- asset ID;
- location;
- inspection date;
- name of Inspector 1;
- name of Inspector 2;
- part ID;
- part type;
- condition (good, moderate, bad, not relevant);
- observed defects.

The template for a time sheet specifies the following data elements:

- contract ID;
- employee name;
- employee ID;
- year;
- week;
- day;
- hours worked;
- description.

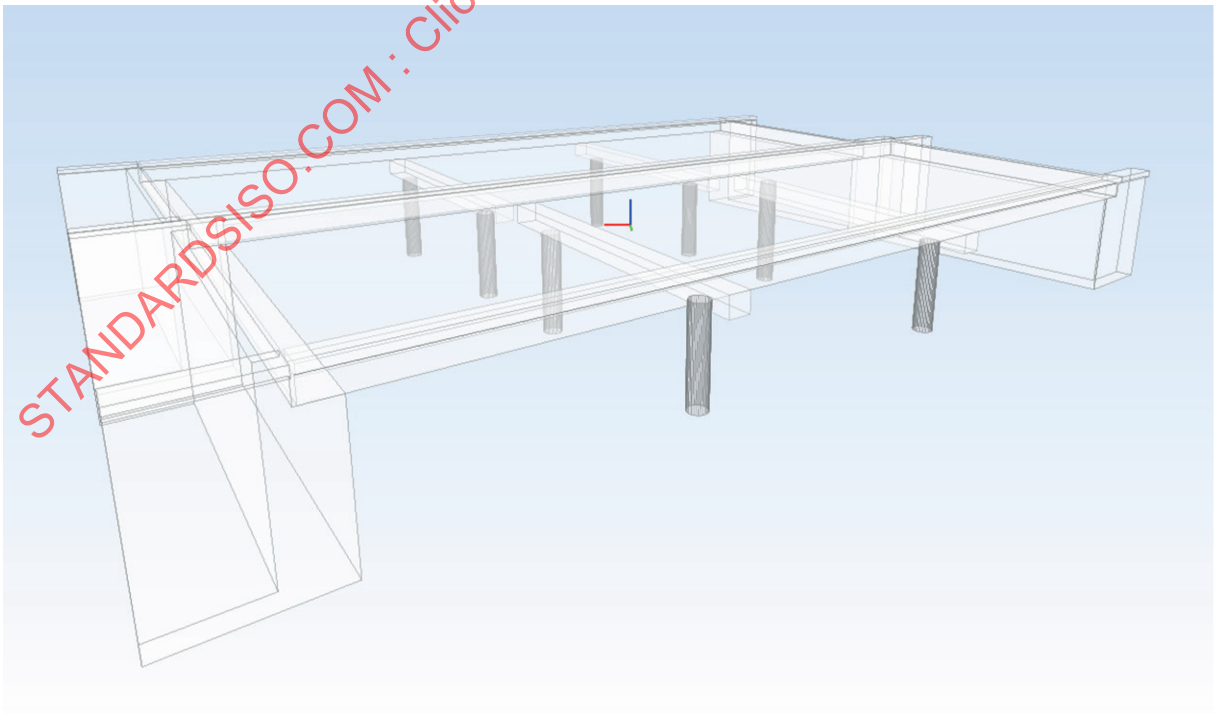
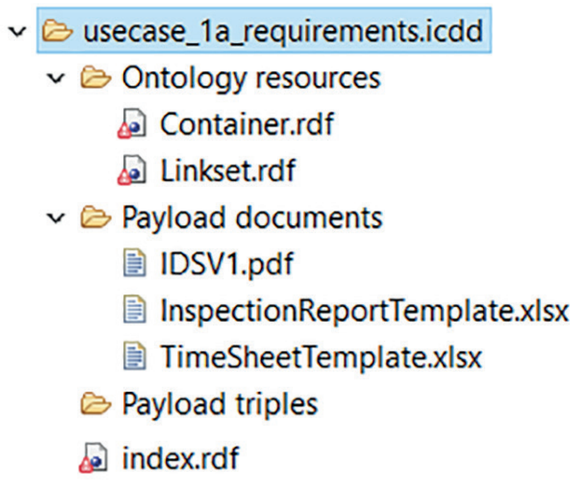


Figure A.1 — Viaduct example

A.2.2 Example container — Information delivery requirements

This example explains the container provided by the client.

Figure A.2 shows the structure of the container and shows the documents that are made available.



usecase_1a_requirements.icdd
- Ontology resources
  - Container.rdf
  - Linkset.rdf
- Payload documents
  - IDSV1.pdf
  - InspectionReportTemplate.xlsx
  - TimeSheetTemplate.xlsx
- Payload triples
- index.rdf

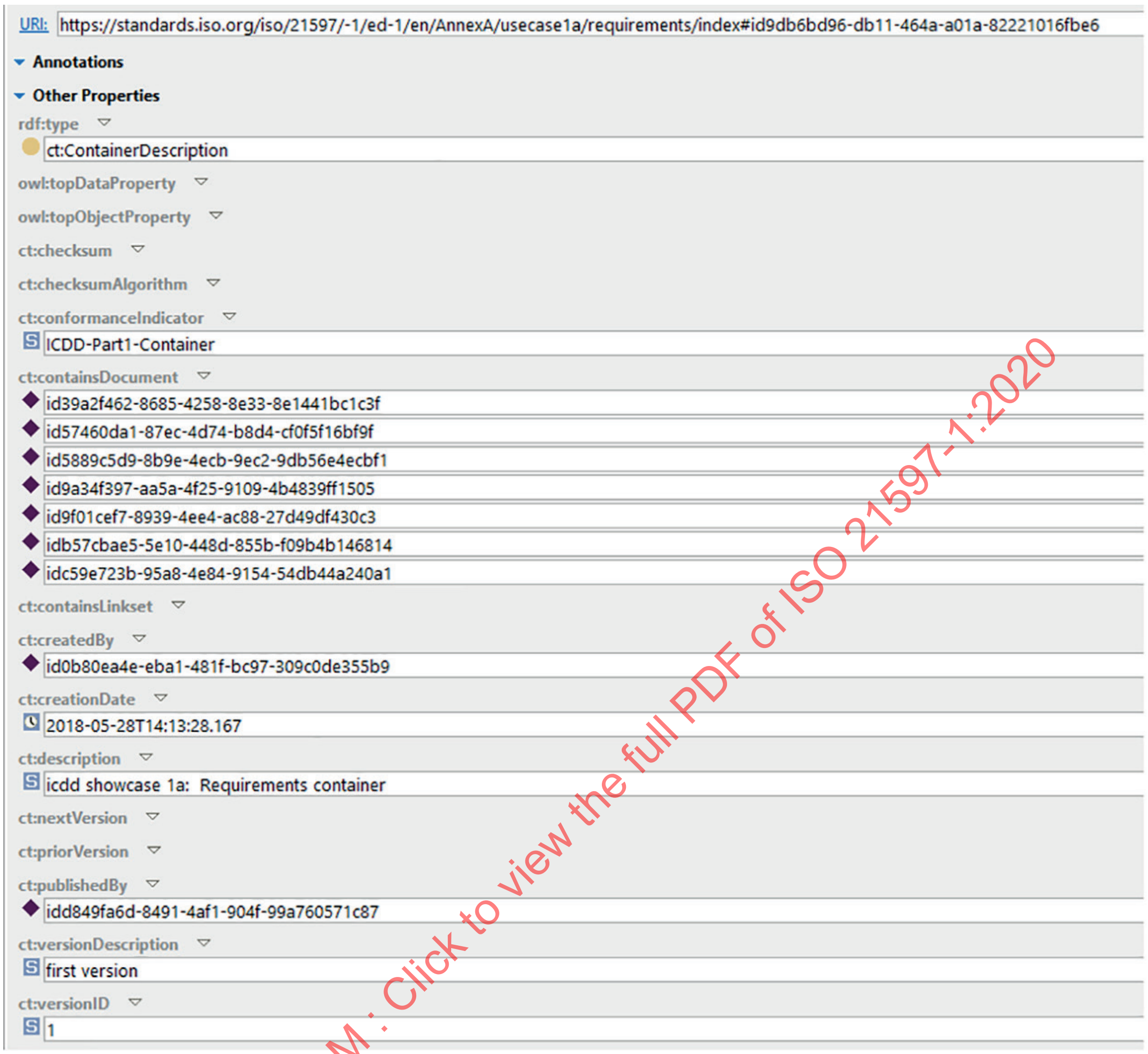
Figure A.2 — Example of a container structure

The Index.rdf in the root folder contains a listing of all the requested documents as well as those already available in the container. Figure A.3 shows an instance of a *ct:ContainerDescription*.

The instance has an URI which is highlighted in the resource form at the top of the screenshot and is treated as the unique identifier for this container. This individual has an *rdf:type* predicate set to *ct:ContainerDescription* making this resource a member of this class. As shown, this individual is described by several other predicates, each either having a prescribed value (e.g. *ct:description*) or referring to URIs that identify other objects (e.g. *ct:createdBy* refers to an instance of *ct:Party* via its URI).

In this example, the individual is related via the *ct:containsDocument* predicate to seven other URIs. One of these is opened up in Figure A.4, showing that it is of the type (*rdf:type*) *ct:InternalDocument* with properties *ct:description*, *ct:filename*, *ct:filetype*, *ct:name* and the *ct:requested* boolean, which in this case, is set to true because the spreadsheet is requested to be provided by the contractor. The other URIs can be opened as well.

Note that some documents are required to be provided by the receiver in a return container (see A.2.3) flagged by setting the *ct:requested* boolean to true. Other documents are available in the container and consequently, are not requested (*ct:requested* is set to false).



URI: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1a/requirements/index#id9db6bd96-db11-464a-a01a-82221016fbe6

Annotations

Other Properties

rdf:type
ct:ContainerDescription

owl:topDataProperty

owl:topObjectProperty

ct:checksum

ct:checksumAlgorithm

ct:conformanceIndicator
ICDD-Part1-Container

ct:containsDocument
id39a2f462-8685-4258-8e33-8e1441bc1c3f
id57460da1-87ec-4d74-b8d4-cf0f5f16bf9f
id5889c5d9-8b9e-4ecb-9ec2-9db56e4ecbf1
id9a34f397-aa5a-4f25-9109-4b4839ff1505
id9f01cef7-8939-4ee4-ac88-27d49df430c3
idb57cbae5-5e10-448d-855b-f09b4b146814
idc59e723b-95a8-4e84-9154-54db44a240a1

ct:containsLinkset

ct:createdBy
id0b80ea4e-eba1-481f-bc97-309c0de355b9

ct:creationDate
2018-05-28T14:13:28.167

ct:description
icdd showcase 1a: Requirements container

ct:nextVersion

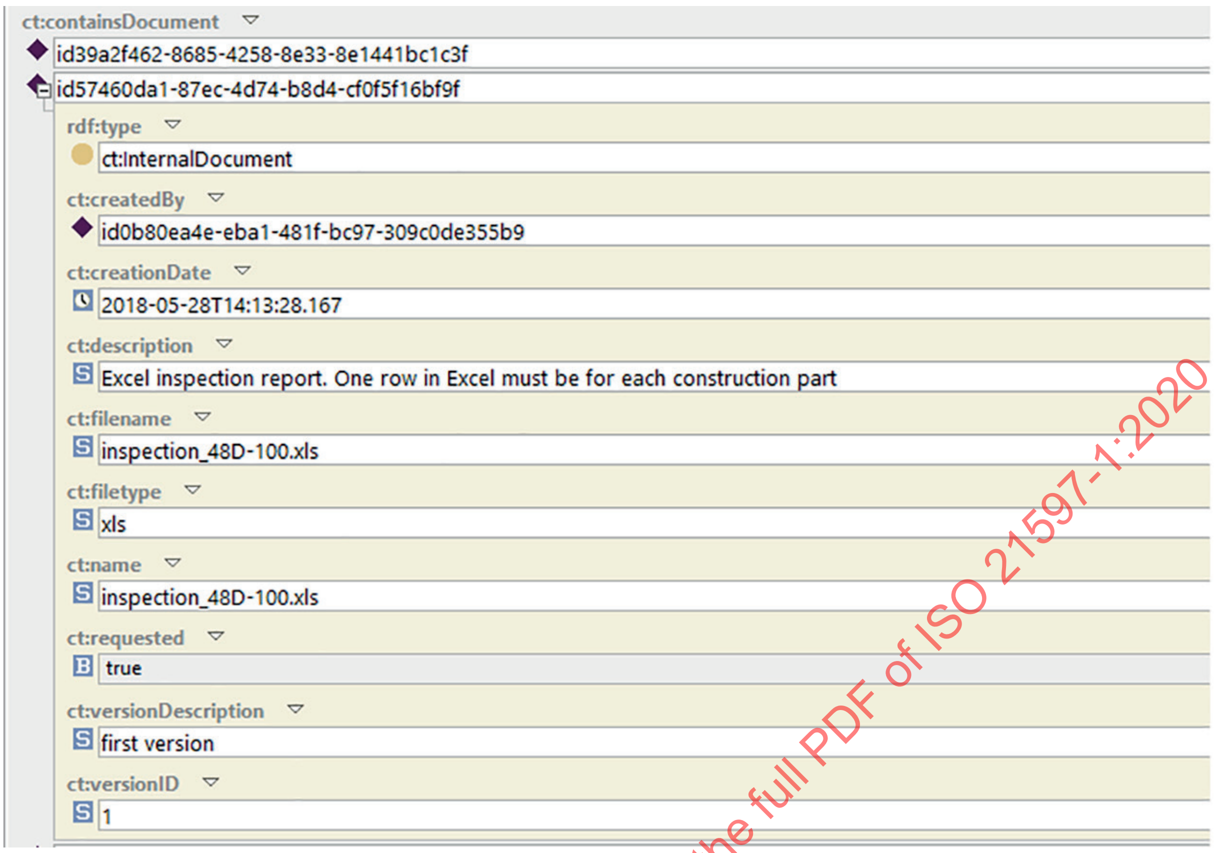
ct:priorVersion

ct:publishedBy
idd849fa6d-8491-4af1-904f-99a760571c87

ct:versionDescription
first version

ct:versionID
1

Figure A.3 — Example of *ct:ContainerDescription* in the Index.rdf file of a container



ct:containsDocument

id39a2f462-8685-4258-8e33-8e1441bc1c3f

id57460da1-87ec-4d74-b8d4-cf0f5f16bf9f

rdf:type

ct:InternalDocument

ct:createdBy

id0b80ea4e-eba1-481f-bc97-309c0de355b9

ct:creationDate

2018-05-28T14:13:28.167

ct:description

Excel inspection report. One row in Excel must be for each construction part

ct:filename

inspection_48D-100.xls

ct:filetype

xls

ct:name

inspection_48D-100.xls

ct:requested

true

ct:versionDescription

first version

ct:versionID

1

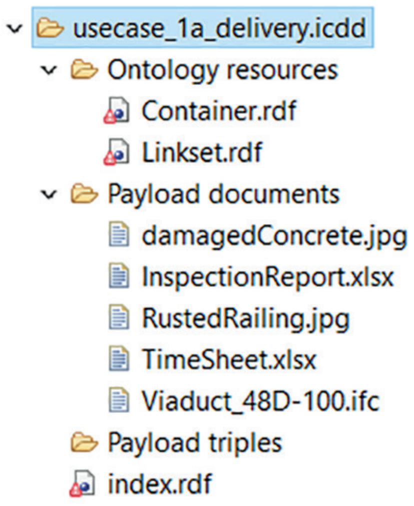
Figure A.4 — A linked document (requested inspection report) in the example Index file

This container can be downloaded via: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1a/requirements.icdd.

A.2.3 Example container — Information delivery

This example explains the container returned by the contractor.

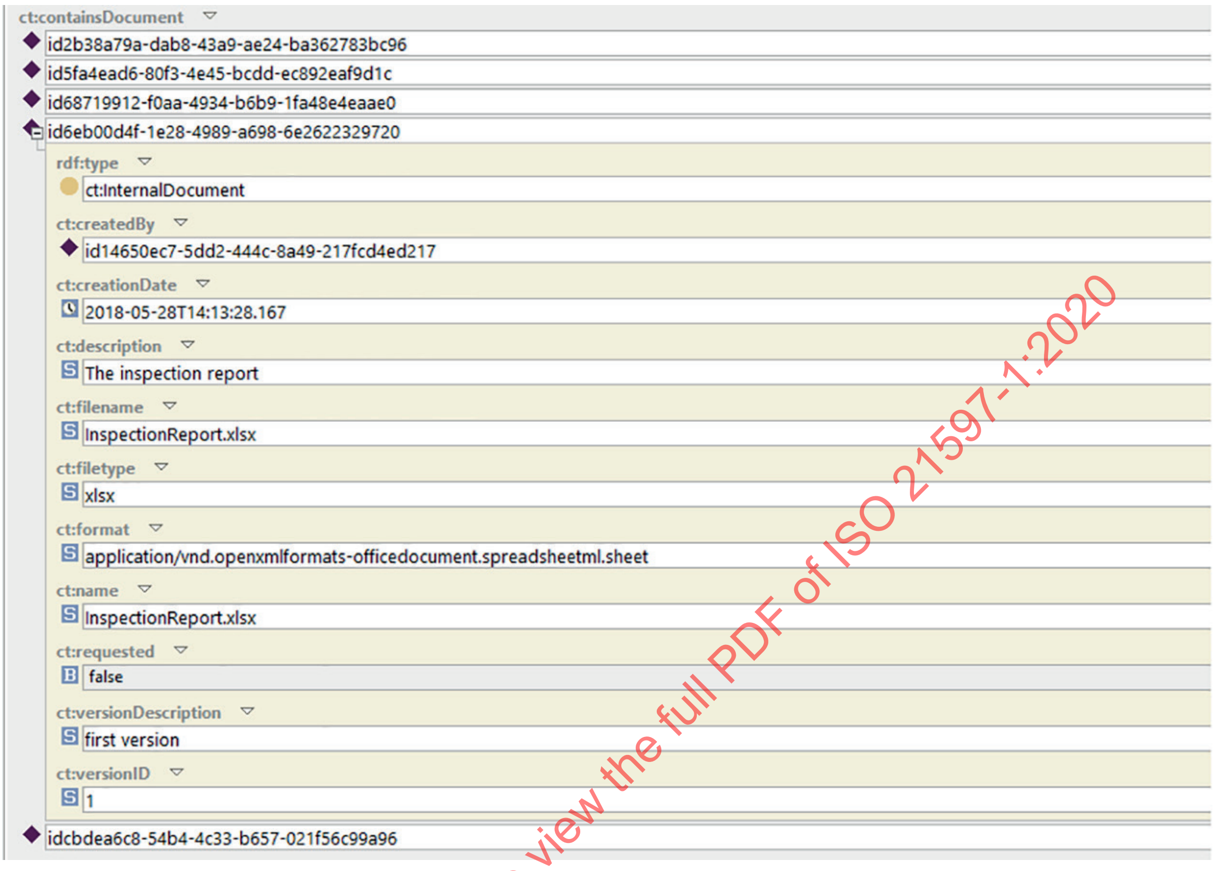
The delivery container contains the inspection report as a spreadsheet and the timesheet in the payload folder of the container. The payload folder also contains a 3D IFC model and some pictures supporting the inspection report. See Figure A.5 for the structure of the container.



```
usecase_1a_delivery.icdd
  Ontology resources
    Container.rdf
    Linkset.rdf
  Payload documents
    damagedConcrete.jpg
    InspectionReport.xlsx
    RustedRailing.jpg
    TimeSheet.xlsx
    Viaduct_48D-100.ifc
  Payload triples
  index.rdf
```

Figure A.5 — Example of container structure

The *ct:ContainerDescription* links to the documents via individuals of *ct:InternalDocuments*. Figure A.6 shows the container description that links to an internal document.



```
ct:containsDocument
◆ id2b38a79a-dab8-43a9-ae24-ba362783bc96
◆ id5fa4ead6-80f3-4e45-bcdd-ec892eaf9d1c
◆ id68719912-f0aa-4934-b6b9-1fa48e4eaae0
◆ id6eb00d4f-1e28-4989-a698-6e2622329720
    rdf:type
    ● ct:InternalDocument
    ct:createdBy
    ◆ id14650ec7-5dd2-444c-8a49-217fcd4ed217
    ct:creationDate
      2018-05-28T14:13:28.167
    ct:description
    S The inspection report
    ct:filename
    S InspectionReport.xlsx
    ct:filetype
    S xlsx
    ct:format
    S application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
    ct:name
    S InspectionReport.xlsx
    ct:requested
    B false
    ct:versionDescription
    S first version
    ct:versionID
    S 1
◆ idcbdea6c8-54b4-4c33-b657-021f56c99a96
```

Figure A.6 — A linked document (inspection report) in the example Index file

This container can be downloaded via: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1a/delivery.icdd.

A.3 Use case 1B — Delivery of documents with links

A.3.1 General

This use case is an extension of use case 1A. In this use case the client asks also to provide

— the links between the construction parts as specified in the inspection report and the IFC elements;

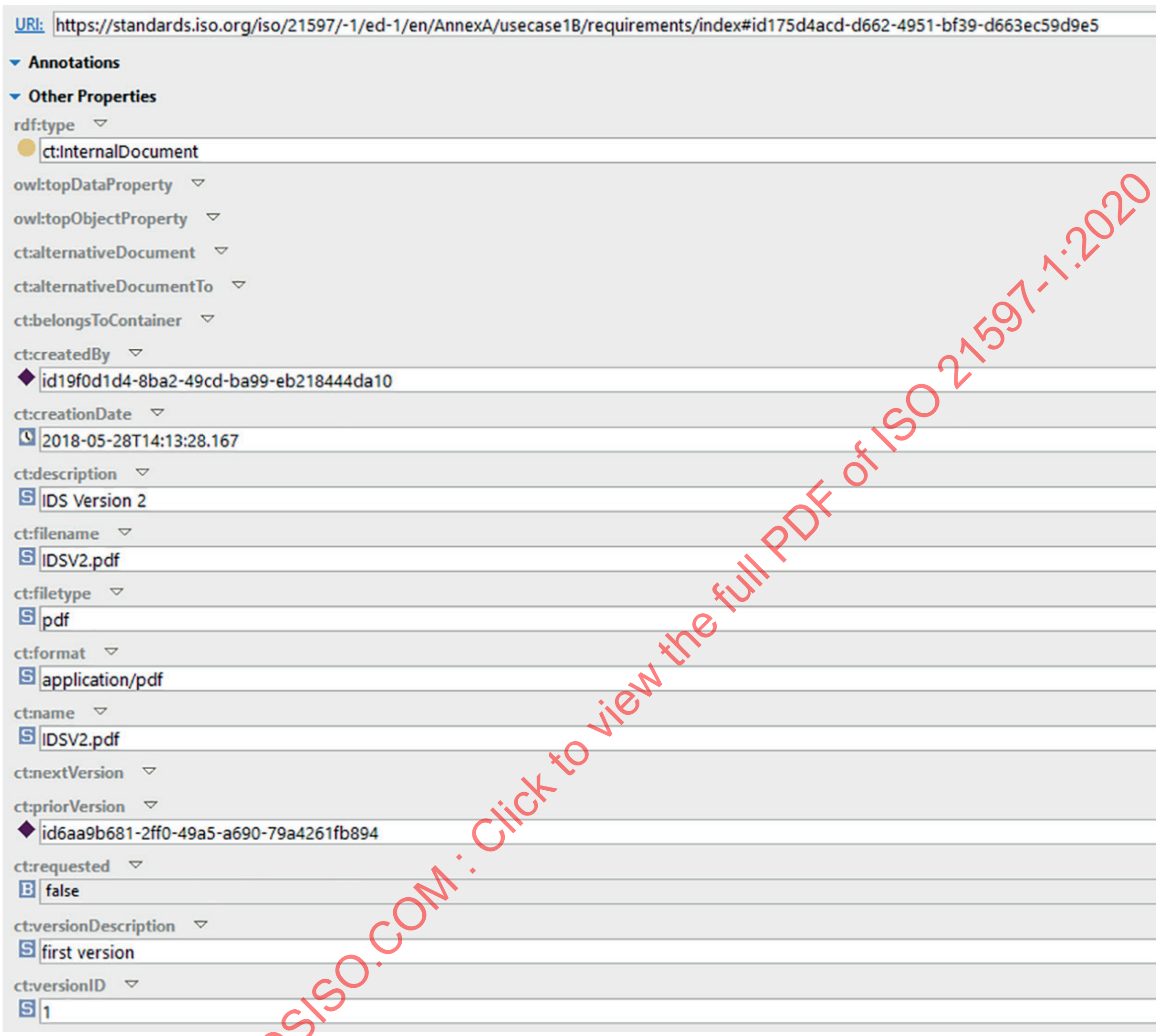
— the links between the construction parts as specified in the inspection report and photos.

For the purpose of the assignment, the client provides a container, equipped with slots for the requested documents. The above-mentioned requirements are added to the information delivery specification and is stored as a successor of the information delivery specification used in the previous use case.

The client uses containers supplied by contractors for the purpose of reviewing inspection reports. This means that the client has an application that shows the inspection report (Excel table), displays the corresponding construction parts in a 3D-model and shows available photos for every construction part.

A.3.2 Example container — Information delivery requirements

This container is very similar to the previous example and contains two information delivery specifications which are related to each other via the *ct:priorVersion* property. Figure A.7 shows the link to the new information delivery specification document.



URI: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1B/requirements/index#id175d4acd-d662-4951-bf39-d663ec59d9e5

Annotations

Other Properties

rdf:type
ct:InternalDocument

owl:topDataProperty

owl:topObjectProperty

ct:alternativeDocument

ct:alternativeDocumentTo

ct:belongsToContainer

ct:createdBy
id19f0d1d4-8ba2-49cd-ba99-eb218444da10

ct:creationDate
2018-05-28T14:13:28.167

ct:description
IDS Version 2

ct:filename
IDSV2.pdf

ct:filetype
pdf

ct:format
application/pdf

ct:name
IDSV2.pdf

ct:nextVersion

ct:priorVersion
id6aa9b681-2ff0-49a5-a690-79a4261fb894

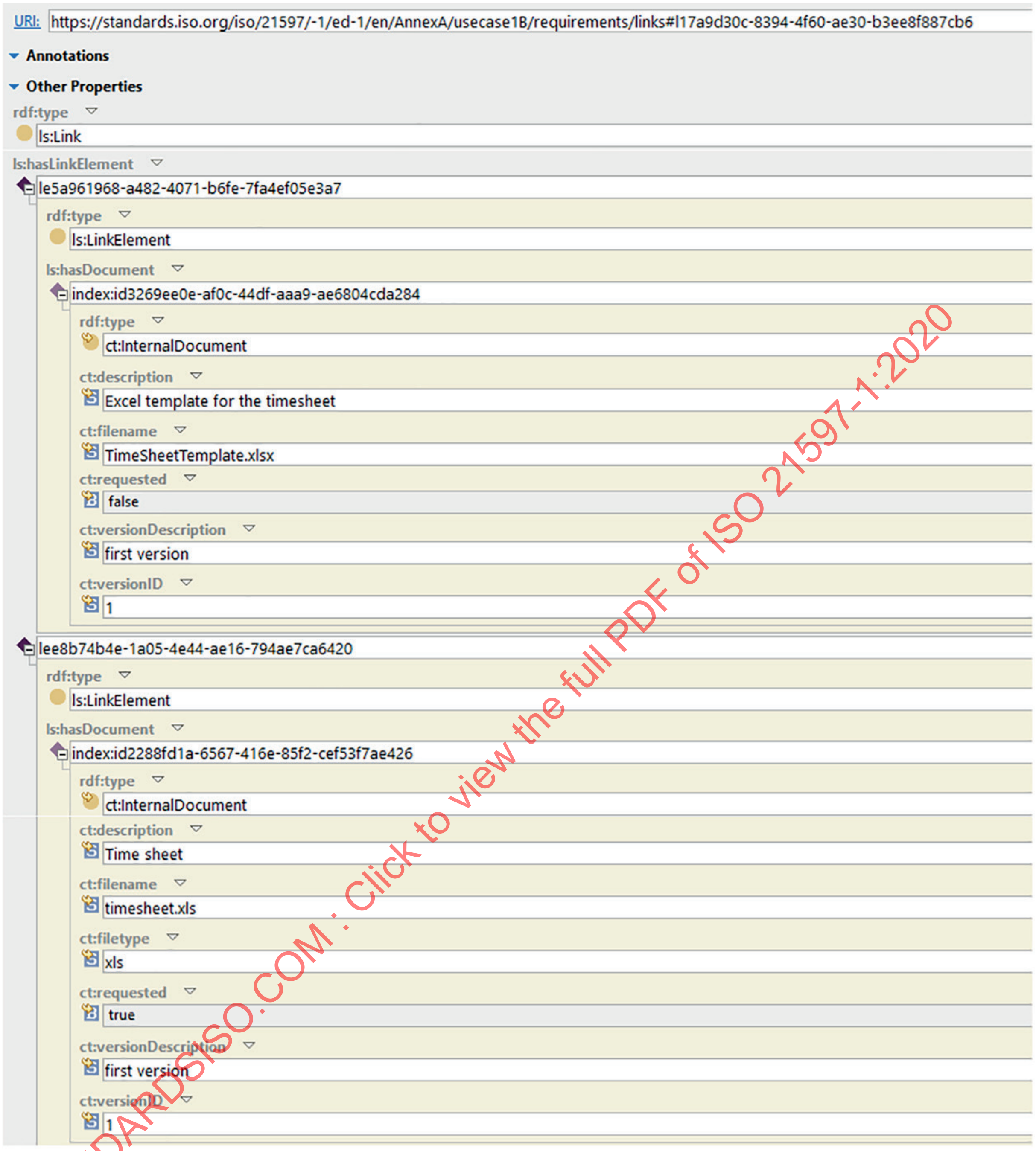
ct:requested
false

ct:versionDescription
first version

ct:versionID
1

Figure A.7 — Example of a new Internal document with a *ct:priorVersion*

In addition, this container contains three links between documents. The links are instances of *ls:Link*. Figure A.8 shows all information of one link. The URI of the *ls:Link* instance is highlighted. This instance is connected via the *ls:hasLinkElement* with 2 instances of *ls:LinkElement* via their respective URIs. Both of these URIs are opened in Figure A.8 and have an *ls:hasDocument* connection to individuals of class *ct:InternalDocument* that are defined in the Index.rdf.



URI: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1B/requirements/links#l17a9d30c-8394-4f60-ae30-b3ee8f887cb6

Annotations

Other Properties

```
rdf:type
  ls:Link
ls:hasLinkElement
  le5a961968-a482-4071-b6fe-7fa4ef05e3a7
    rdf:type
      ls:LinkElement
    ls:hasDocument
      index:id3269ee0e-af0c-44df-aaa9-ae6804cda284
        rdf:type
          ct:InternalDocument
        ct:description
          Excel template for the timesheet
        ct:filename
          TimeSheetTemplate.xlsx
        ct:requested
          false
        ct:versionDescription
          first version
        ct:versionID
          1
  lee8b74b4e-1a05-4e44-ae16-794ae7ca6420
    rdf:type
      ls:LinkElement
    ls:hasDocument
      index:id2288fd1a-6567-416e-85f2-cef53f7ae426
        rdf:type
          ct:InternalDocument
        ct:description
          Time sheet
        ct:filename
          timesheet.xls
        ct:filetype
          xls
        ct:requested
          true
        ct:versionDescription
          first version
        ct:versionID
          1
```

Figure A.8 — Example of a link between a Timesheet template and the requested timesheet

This container can be downloaded via: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1b/requirements.icdd.

A.3.3 Example container — Information delivery

This example explains the container returned by the contractor.

The contractor provides a container with the following information:

— inspection report;

— timesheet report;

— IFC model;

— photos of detected asset issues;

— the links between the construction parts as specified in the Inspection report and the IFC elements;

— the links between the construction parts and photos.

The container has detailed links between the IFC model and the inspection report spreadsheet. Elements in the IFC file are identified using the GUID and linked to a specific row in the inspection report. In addition, the link includes a picture. Consequently, an *ls:Link* is related to three instances of *ls:LinkElement*. Figure A.9 shows an *ls:Link* instance that has three instances of *ls:LinkElement*. One of these instances is opened, which refers via *ct:Linkset:hasDocument* to an internal document (rustedRailing.jpg).

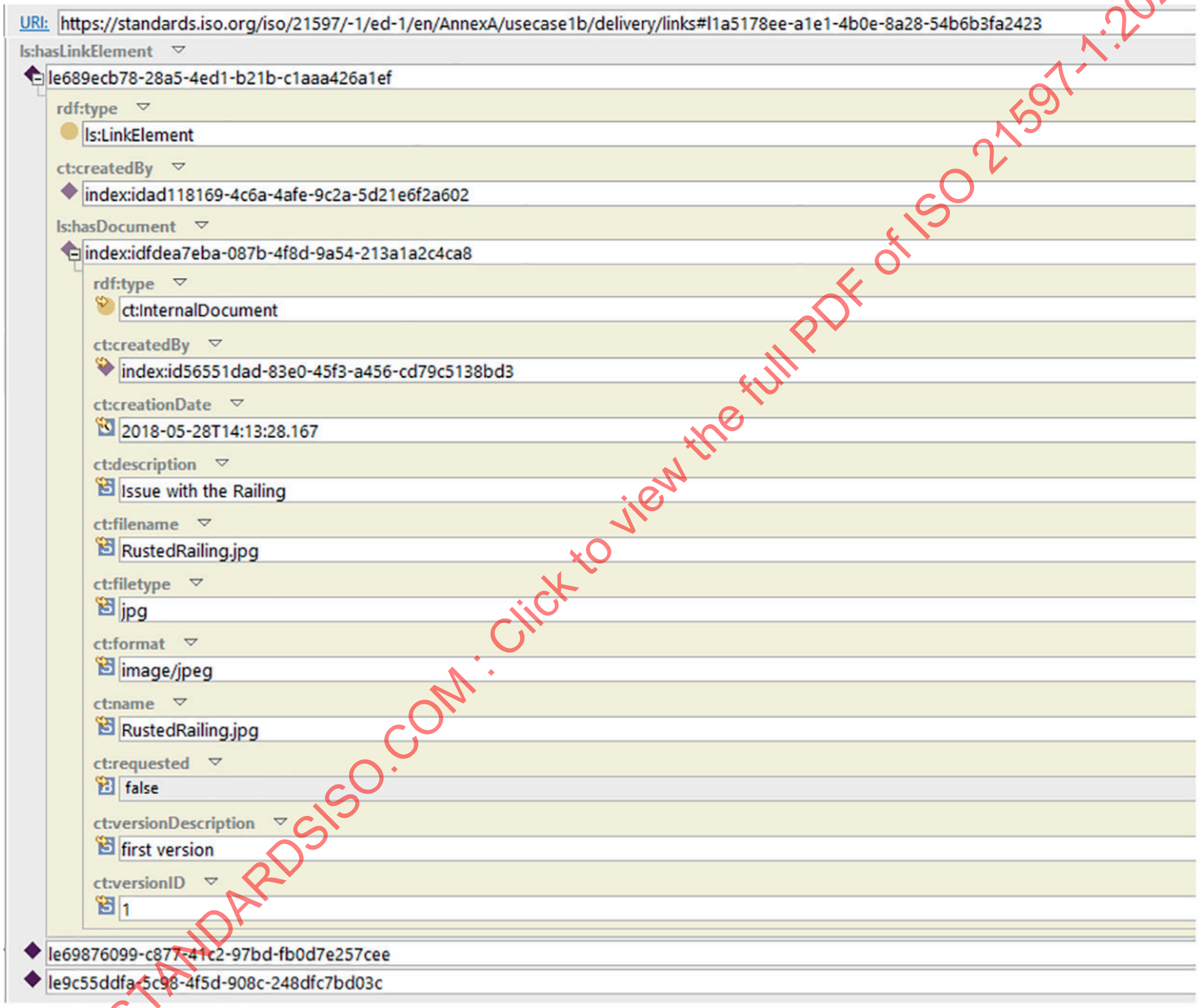

```
URI: https://standards.iso.org/iso/21597/-1/ed-1/en/AnnexA/usecase1b/delivery/links#l1a5178ee-a1e1-4b0e-8a28-54b6b3fa2423
ls:hasLinkElement
  le689ecb78-28a5-4ed1-b21b-c1aaa426a1ef
    rdf:type
      ls:LinkElement
    ct:createdBy
      index:idad118169-4c6a-4afe-9c2a-5d21e6f2a602
    ls:hasDocument
      index:idfdea7eba-087b-4f8d-9a54-213a1a2c4ca8
        rdf:type
          ct:InternalDocument
        ct:createdBy
          index:id56551dad-83e0-45f3-a456-cd79c5138bd3
        ct:creationDate
          2018-05-28T14:13:28.167
        ct:description
          Issue with the Railing
        ct:filename
          RustedRailing.jpg
        ct:filetype
          jpg
        ct:format
          image/jpeg
        ct:name
          RustedRailing.jpg
        ct:requested
          false
        ct:versionDescription
          first version
        ct:versionID
          1
  le69876099-c877-41c2-97bd-fb0d7e257cee
  le9c55ddfa-5c98-4f5d-908c-248dfc7bd03c
```

Figure A.9 — Example of a link with an opened *ls:LinkElement* pointing to a JPG file

The other instances of *ls:LinkElement* refer to the excel sheet and to an IFC Entity within an IFC file. Therefore an *ls:StringBasedIdentifier* is used to capture the GUID of the IFC entity. Figure A.10 shows the same *ls:Link* but this time the *ls:LinkElement* referring to the inspection report and the *ls:LinkElement* referring to the IFC file are opened. Also the *ct:Linkset:hasIdentifier* is opened demonstrating the reference to an *ls:StringBasedIdentifier*.