INTERNATIONAL STANDARD

ISO/IEC 19773

First edition
2011-09-01

Information technology — Metadata Registries (MDR) modules

Technologies de l'information — Modules de registres de métadonnées (MDR)

Reference number
ISO/IEC 19773:2011(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 19773 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

Introduction

This International Standard specifies small modules of data that can be used or reused in applications. These modules have been extracted from ISO/IEC 11179-3, ISO/IEC 19763, and OASIS EBXML, and have been refined further. These modules are intended to harmonize with current and future versions of the ISO/IEC 11179 series and the ISO/IEC 19763 series.

During the development of this International Standard, it was originally presented as a multipart standard consisting of an overview part and other parts, one for each module. However, this presentation approach proved to be too cumbersome for users, with some duplication of text and cross-references across multiple documents. The work was consolidated into a single document that facilitated ongoing additions and amendments, as industry and technology demand.

In the present version of this International Standard, subclauses of Clause 3 and Clause 9 itself are marked “reserved for future use”. Future amendments might insert text into these (currently) reserved areas. Meanwhile, the document as a whole is designed with a parallel structure (terminology in Subclause 3.*X* corresponds to the data structure in Clause *X*), so that the user can quickly locate module-specific terminology for a module-specific data structure. Thus, for the UPU postal data module, the terminology is defined in Subclause 3.*16* and its corresponding data structure is described in Clause *16*.

Information technology — Metadata Registries (MDR) modules

1 Scope

This International Standard specifies the technical interoperability details of metadata modules, which are used in ISO/IEC 11179.

2 Normative references

The following referenced documents are indispensable for the application 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 11404:2007, *Information technology — General-Purpose Datatypes (GPD)*

ISO/IEC 20944-1:—, *Information technology — Metadata Registries Interoperability and Bindings (MDR-IB) — Part 1: Framework, common vocabulary, and common provisions for conformance*

ISO 21090:2011, *Health informatics — Harmonized data types for information interchange*

IETF RFC 2421, *Voice Profile for Internet Mail — Version 2,* September 1998

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax,* January 2005

IETF RFC 3987, *Internationalized Resource Identifiers (IRIs),* January 2005

IETF RFC 5646, *Tags for Identifying Languages,* September 2009

UPU S42a-6:2009, *International postal address components and templates — Part A: Conceptual hierarchy and template languages*[1)]

3 Terms, definitions, and abbreviations

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

3.1 Signifiers, referencing, and their associations

3.1.1
to reference, verb
to create an association with a particular object

[ISO/IEC 20944-1]

EXAMPLE A human pointing at an object.

1) UPU is the Universal Postal Union at http://www.upu.int. UPU S42a-6 is based on EN 14142-1, *Postal services — Address databases — Part 1 — Components of postal addresses*.

3.1.2
reference, noun
association with a particular object (the referent)

[ISO/IEC 20944-1]

NOTE In one context, a *reference* is the opposite of a *literal*: the literal gives the data at hand, while the reference points to the data, which must be subsequently accessed, retrieved, or written.

EXAMPLE An association created by proximity; a computer memory pointer; a database foreign key.

3.1.3
referent, noun
object that is referenced

[ISO/IEC 20944-1]

3.1.4
to dereference
to access the referenced object (the referent)

[ISO/IEC 20944-1]

3.1.5
signifier
sign, noun
general concept, whose extension is perceivable objects that are associated with objects

NOTE A signifier associated via a designating relationship to an object of the variety “concept” is known as a signifier designating a concept, i.e., a designation.

3.1.6
label
reference to an object by a signifier

3.1.7
designation
representation of a concept by a signifier which denotes it

NOTE Adapted from ISO 1087-1, where the word “sign” is used. In this context, both “sign” and “signifier” mean the same, but the word “signifier” is used to avoid confusion with other senses of the word “sign”.

3.1.8
identifier
label that is intended to be dereferenced

[ISO/IEC 20944-1]

NOTE 1 An identifier is also a reference.

NOTE 2 This definition is consistent with IETF RFC 3986, which describes the Uniform Resource Identifier (URI) syntax and semantics, and IETF RFC 3987, which describes the Internationalized Resource Identifier (IRI) syntax.

3.1.9
literal
lexical token that, from a syntactic point of view, stands for itself

[ISO/IEC 2382-5]

NOTE In this context, a *literal* is the opposite of a *reference*: the lexical token for a reference does not stand for itself, but standards for something else.

EXAMPLE The names **JANUARY**, **FEBRUARY**, **MARCH**, etc. in the following definition of a datatype are literals:

```
Month_Type is (JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE,
        JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER);
Month : Month_Type;
.....
Month := APRIL;
```

The token **JANUARY** stands for itself (meaning the first month of the year) in contrast to a token **JANUARY** representing a variable that is unconstrained and can stand for other values.

3.1.10
namespace
set of labels with a defined scope of usage

[ISO/IEC 20944-1]

3.1.11
namespace label
label that designates a namespace

[ISO/IEC 20944-1]

3.2 Fundamental datatypes

3.2.1
string
sequence of elements of the same nature, such as characters or bits, considered as a whole

[ISO/IEC 2382-4:1999]

NOTE A string may be empty or contain only one element.

3.2.2
character string
string consisting solely of characters

[ISO/IEC 2382-4:1999]

3.2.3
text
data in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language

[ISO/IEC 2382-23:1994]

EXAMPLE A business letter printed on paper or displayed on a screen.

3.2.4
datatype
set of distinct values, characterized by properties of those values, and by operations on those values

[ISO/IEC 11404]

3.2.5
characterizing operations (of a datatype)
collection of operations on, or yielding, values of the datatype that distinguish this datatype from other datatypes with identical value spaces

[ISO/IEC 11404]

3.2.6
value
object whose totality can be used for computation

NOTE In contrast to an object (used in the sense of object-oriented computing), the totality and boundary of a value is known and determinate, whereas the totality or boundary of an (object-oriented) object can be unknown or indeterminate.

3.2.7
value space
set of values for a given datatype

[ISO/IEC 11404]

3.3 Generic implementation-related concepts

3.3.1
execution time
run time
any instant at which the execution of a particular program takes place

[ISO/IEC 2382-7]

3.3.2
context of use
users, tasks, equipment (hardware, software, and materials), and the physical and social environments in which a product, process, or service is used

NOTE Adapted from ISO 9241-11:1998.

3.3.3
smallest permitted maximum
SPM
⟨technical specification⟩ meet-or-exceed provision that specifies a required minimum value for an implementation value describing the maximum value of a sizing parameter

[ISO/IEC 20944-1]

EXAMPLE In the provision “the smallest permitted maximum length of field **X** shall be **17**”, the SPM is **17** (which applies to the field length). This provision means: implementers may implement “field **X**” with a maximum length of **17**, **18**, **19**, etc., but not with a length of **16** or less. Thus, `char X[17]` satisfies the implementation requirements, even though the data itself might be smaller, e.g., a 10-character string stored in a 17-character array.

NOTE An SPM sets a lower bound for an implementation-defined maximum value.

3.3.4
implementation-defined, adj
⟨technical specification⟩ unspecified, yet each implementation documents how the choice among the available alternatives is made

[ISO/IEC 20944-1]

EXAMPLE 1 An implementation-defined feature; an implementation-defined value; an implementation-defined behavior.

EXAMPLE 2 A standard specifies that size of array **X** is implementation-defined with a minimum size of **17**. This provision implies two requirements: (1) the size of the array is greater than or equal to **17**, and (2) the implementation will document the actual size. This example is a meet-or-exceed provision (e.g., a smallest permitted maximum).

NOTE The distinction between "unspecified" and "implementation-defined" is that the latter requires implementation documentation while the former does not require implementation documentation (nor does the former prohibit implementation documentation).

3.4 Terminology applicable to more than one module

3.4.1
metadata module
metadata registries module
unit or grouping of descriptive data which is created, managed, used, or interchanged among autonomous parties as a single discrete set in a specified context

3.5 Reserved for future use

3.6 Reserved for future use

3.7 Reserved for future use

3.8 Reserved for future use

3.9 Reserved for future use

3.10 Module 10-specific terminology: Data structure for reference-or-literal (reflit)

3.10.1
characterstring
ISO/IEC 11404 datatype for representing character strings

NOTE The ISO/IEC 11404 **`characterstring`** datatype takes the parameter **`repertoire`** that indicates the logical set of characters. Typically, **`characterstring(iso-10646-1)`** is be used to portably store text data, i.e., its value will be preserved across all implementations of the datatype.

3.10.2
octet string
string consisting solely of octets

3.10.3
octetstring
ISO/IEC 11404 datatype for representing octet strings

NOTE An **`octetstring`** datatype can be used to portably store binary data, i.e., its value will be preserved across all implementations of the datatype.

3.10.4
reflit
datatype whose value can be accessed directly as a literal value or accessed indirectly via a reference to a value

3.11 Module 11-specific terminology: Data structure for multiple internationalized/localized values and data

3.11.1
multidata
set of values that all have the same meaning, but may have different presentations and different datatypes that are dependent upon the context of use

3.11.2
multivalue
set of values of a defined datatype that all have the same meaning, but may have different presentations dependent upon context of use

NOTE A multivalue is a multidata for a defined datatype.

3.12 Module 12-specific terminology: Data structure for multiple internationalized/localized strings and texts

3.12.1
document
named, structured unit of text and possibly images that can be stored, edited, retrieved, and exchanged among systems or users as a separate unit

3.12.2
multistring
set of character strings that all have the same meaning, but may have different presentations dependent upon context of use

NOTE 1 A multistring is similar to a multivalue except that the multistring is specialized for character strings.

NOTE 2 A multistring is similar to a multitext except that the multitext may present textual information in a form other than a string, such as a document or an image.

3.12.3
multitext
set of textual values that all have the same meaning, but may have different presentations and different datatypes that are dependent upon the context of use

3.13 Module 13-specific terminology: Data structure for slot tuple

3.13.1
slot tuple
3-tuple comprised of a identifier, a kind, and a value

3.13.2
slot_tuple
datatype for the slot tuple

3.13.3
kind
⟨data classification⟩ category within a classification system

NOTE 1 Typically, a classification system classifies *extensions*, *intensions*, or both. Within this note, the terms *extension* (totality of objects to which a concept corresponds) and *intension* (set of characteristics which makes up a concept) correspond to their use in the field of terminology (see ISO 1087-1).

NOTE 2 A *datatype* is a specialization of a *kind* (i.e., "a kind of a kind"). A *datatype* differs from a *kind* in the following ways: (1) a *datatype* implies a *value space*, which is a specialization of an *extension* (totality of values vs. totality of objects), (2) a *datatype* implies *characterization by properties* and *characterizing operations*, which is a specialization of an *intension* (data-like properties and characterizing operations vs. characteristics in general), and (3) a *datatype* implies both a defined *extension* and defined *intension*, whereas a *kind* does not require both an *extension* and an *intension*.

NOTE 3 The term *kind* (a characteristic of an object) should not be confused with *kind of* (an indication of a general concept in a generic-specific relationship, e.g., a dog is a *kind of* mammal).

3.13.4
N-tuple
tuple of length N

EXAMPLE The tuple **<x,y,z>** is a 3-tuple; the tuple **<x,y>** is a 2-tuple (also known as a pair); the tuple **<x>** is a 1-tuple (note that **<x>** ≠ **x** because **<x>** is a list and **x** is a scalar); and the tuple **<>** is a 0-type (also known in mathematics as the "unit type").

3.13.5
tuple
ordered list of elements

EXAMPLE The tuple **<x,y,z>** contains the three values **x**, **y**, and **z**; and **<x,y,z>** is not equal to **<z,y,x>**. In contrast to the set (whose elements are unordered), the set **{x,y,z}** is equal to the set **{z,y,x}**.

3.14 Module 14-specific terminology: Data structure for unstructured table of slot tuples

There are no additional definitions for Module 14.

3.15 Module 15-specific terminology: Data structure for reified relationships and relationships systems

3.15.1
relationship
association among zero or more objects with defined roles for each object

3.15.2
relationship system
set of objects and their relationships

3.16 Module 16-specific terminology: Data structure for UPU postal data

3.16.1 Terminology from UPU S42a-6

3.16.1.1
addressee
party who is the intended ultimate recipient of a postal item

[UPU S42a-6]

NOTE 1 The addressee may be explicitly defined as part of the postal address, or may be implicit. For example, in certain countries, omission of addressee information may be taken as implying that delivery must be to an individual or legal entity having legal access to the delivery point.

NOTE 2 An address may contain multiple addressee specifications. For example, Mr. or Mrs. Smith specifies that the addressee is either one of two individuals, whilst Mr. Jones and Mrs. Smith denotes that the addressee is a group of two individuals. See also addressee role descriptor.

NOTE 3 The above definition differs from that in ENV 13712.

NOTE 4 The use made by the postal operator of addressee and mailee data might be dependent on the postal service applicable to the postal item. For certain services, such as registered mail, the postal operator's responsibility might include ensuring that the addressee, or a duly authorised representative, acknowledges receipt of the postal item. In other cases, addressee data could be purely informative or used by the postal operator only for consistency checking and/or for the activation of forwarding services. In still other cases, it might be used for sorting or sequencing purposes prior to delivery (e.g. in the case of business mail being pre-sequenced by department or individual company official).

NOTE 5 When the addressee is explicitly defined (see NOTE 1), there is always one addressee in a syntactically correct postal address, whereas the mailee information does not have to be present. In some countries, the addressee may be an abstraction such as "Postal Customer".

3.16.1.2
delivery
postal process in which a postal item leaves the responsibility of the postal operator through being handed over to, or left for collection by, the addressee, the mailee or an authorized representative, or deposited in a private letter box accessible to one or other of these

[UPU S42a-6]

NOTE Except in the case of special services, for which the addressee or mailee is required to acknowledge receipt, delivery does not necessarily guarantee that the postal item actually reaches the addressee or mailee. In particular, where postal items are left for collection or deposited in a private letter box, other persons may have access to them, either legally or otherwise.

3.16.1.3
delivery address
postal address specified by the mailer to which the postal operator is requested to deliver the postal item

[UPU S42a-6]

NOTE 1 The delivery address may in certain circumstances, e.g. unaddressed mail, not actually be represented on the postal item. In this case, the delivery address is determined by the postal operator in accordance with an agreement between the operator and the mailer.

NOTE 2 The postal item may not actually be delivered to the requested delivery address. For example, in the case of forwarding, delivery takes place at the forwarding address.

3.16.1.4
delivery point
physical location recognized by a postal operator as a valid location at which delivery of a postal item may occur

[UPU S42a-6]

3.16.1.5
forwarding address
postal address, specified by the addressee or mailee of a postal item, to which the postal operator is requested to deliver the postal item, in place of delivering it to the delivery address

[UPU S42a-6]

NOTE 1 Not all postal items can be forwarded, as for some postal services the mailer might require the return of the postal item if it cannot be delivered at the delivery address.

NOTE 2 Forwarding addresses can be permanent, e.g. in case of relocation of the addressee, or temporary. They may also involve the holding of mail for collection by the addressee or the mailee (see poste restante).

3.16.1.6
mail recipient
individual who actually receives a postal item at delivery, or who first accesses the postal item if it is left for collection

[UPU S42a-6]

NOTE The mail recipient should normally be the addressee, the mailee or an authorized representative of one of these two. However, this may not always be the case, e.g. if the postal item is left for collection in a location to which third parties have access; if the addressee/mailee have moved without leaving forwarding instructions, or if the addressee or mailee specification was ambiguous and was, as a result, misinterpreted by the postal operator.

3.16.1.7
mail submitter
party responsible for induction of a postal item into the postal system

[UPU S42a-6]

NOTE The mail submitter may be, but is not necessarily, the same party as the mail originator.

3.16.1.8
mailee
party designated in a postal address as having responsibility for ensuring that postal items, delivered or handed over by the postal operator at the delivery address, reach their addressee

[UPU S42a-6]

NOTE 1 Unlike the addressee, mailee is never implicit: if a postal address does not contain a mailee specification, then there is no mailee.

NOTE 2 Notwithstanding NOTE 1, the mailee may be designated explicitly by use of a role descriptor, or designated implicitly with no role descriptor.

NOTE 3 As is the case for addressee, the mailee specification may be ambiguous.

3.16.1.9
mailer
natural or legal person designated in a postal address as having responsibility for ensuring that postal items, delivered or handed over by the postal operator at the delivery address, reach their addressee

[UPU S42a-6]

NOTE Many processes are involved in the production and mailing of postal items. These include: initiation; content production, which might be separated into parts produced by several different parties (e.g. inserts might be produced separately from covering letters); finishing, including assembly of the content and its packaging (e.g. placing in an envelope, or wrapping) for mailing purposes; addressing; induction into the postal system; payment.

These processes may be performed by one party, or may be split between different parties, each fulfilling a particular role or combination of roles. Where it is necessary to distinguish between such roles, they are referred to by separate terms, in particular mail originator, mail submitter and payer; where such distinction is not necessary, mailer is used as a generic term.

3.16.1.10
party
one or more natural and/or legal persons and/or organisations without legal personality that act(s) as a single entity for the purpose of participation in a transaction associated with a postal item

[UPU S42a-6]

3.16.1.11
postal address
set of information which, for a postal item, allows the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee and/or a mailee

[UPU S42a-6]

NOTE Postal addresses may be ambiguous, incorrect or non-existing. See also syntactically correct postal address, valid postal address.

3.16.1.12
postal address component
collective term for postal address elements, postal address constructs and postal address segments

[UPU S42a-6]

NOTE Clause 16 defines the postal address components which may occur in an actual postal address. Not all components are necessarily used in a specific instance or class of postal addresses.

3.16.1.13
postal address construct
combination of postal address elements which together form a logical portion of a postal address

[UPU S42a-6]

NOTE 1 Constructs can be defined hierarchically. That is, a construct can comprise a logical grouping of postal address elements, a logical grouping of lower level constructs, or a combination of elements and lower level constructs.

NOTE 2 Not all constructs are necessarily used in a specific instance or class of postal addresses.

3.16.1.14
postal address element
basic entity of a postal address that has a well-defined conceptual meaning and representation and has significance for customer or postal processing purposes

[UPU S42a-6]

NOTE 1 A thoroughfare name which may comprise one or more words is an example of a postal address construct, but that does not imply that the individual words of which it is comprised are also constructs. For example, with Pine Grove Avenue, there are at most two postal address constructs. So Pine Grove might be considered as a postal address element, the thoroughfare name. On the other hand, it is part of a larger thoroughfare construct that includes thoroughfare type and thoroughfare qualifier. And these entities can precede or follow the thoroughfare name. This makes it helpful to have separate placeholders for each possible sequential ordering of components in designing postal address templates, and since the meaning of an element is independent of the position, this shows the need for element sub-types alongside elements.

So is thoroughfare name an element sub-type, an element, or a larger construct made up of elements? S42 approaches this by defining those components needed to represent instances or parts of constructs as element sub-types. After that, the elements are the lowest level constructs remaining. As a result, some elements have one or more levels of sub-types, while others have none. The remaining components above the element level are the higher level constructs and segments.

Alternate representations of information that have a distinct function are given the status of elements, which conforms with the above definition of postal address element. An example would be country name and ISO 3166-1 country code, which are separate elements.

On the segment level, though not the construct level. It is possible to replicate a group of elements and have them recognized in the templates. This provides a way to solve certain problems in designing address database, such as multiple addressees at one address, or multiple addresses for one addressee.

Leaving aside the cases of representations and replication, S42 handles multiplicity and subdivision of elements by defining element sub-types. It uses two levels of sub-type in the notation, one for instances and one for parts. Instances can be levels, positions, or occurrences, and parts can be physical or logical. This approach keeps the number of postal address elements limited. Elements should have meaning in a general rather than only a specialized postal context, while this is not always the case with element sub-types, particularly those representing parts of elements. Some cases could be decided either way, but this approach results in combining some previously defined elements, including the components of thoroughfare and the components of delivery service identifier, into single elements, while leaving others such as surname prefix and name qualifier to retain their status as elements.

NOTE 2 Not all elements are necessarily used in a specific instance or class of postal addresses.

3.16.1.15
postal address element and element sub-type code
alternate representation for a postal address element or element sub-type which uses a condensed notation that conforms to specified conventions, is suitable for use in templates, and is relatively language independent when compared with the element and element sub-type names

[UPU S42a-6]

3.16.1.16
postal address element sub-type
sub-division of a postal address element representing parts or instances of the root element, used to facilitate template design, address rendition, address database storage and related technical needs

[UPU S42a-6]

3.16.1.17
postal address segment
named group of related postal address constructs and/or postal address elements

[UPU S42a-6]

3.16.1.18
postal address structure
manner in which postal address components are or can be combined to form a postal address

[UPU S42a-6]

NOTE Postal address structures may differ from country to country, from region to region or even from operator to operator within a country.

3.16.1.19
postal address template
statement of how a postal address is to be written; in particular, it shows the order in which postal address elements are to appear, distinguishes between mandatory and optional postal address elements and provides rendition instructions

[UPU S42a-6]

3.16.1.20
poste restant
delivery service indicator specifying that a postal item is to be held at a designated postal establishment or agency for collection by the addressee or his/her authorised representative

[UPU S42a-6]

3.16.1.21
postal data identifier
alphanumeric prefix to a data structure that defines the content, format and intended interpretation of the data

[UPU S42a-6]

3.16.1.22
rendition instruction
definition of how address elements shall be formatted, or in some cases optionally may be formatted, when printed on a mail piece

[adapted from UPU S42a-6]

NOTE Rendition instructions reflect rules for properly formatting addresses, including punctuation, spacing, fonts, the format of the postcode, locations for identifying marks and codes, abbreviations, and techniques for shortening and reorganizing components to ensure deliverability when there are constraints on available label space.

3.16.1.23
return address
postal address to which the postal operator should deliver a postal item if it is unable to effect normal delivery to the delivery address or, if specified, a forwarding address

[UPU S42a-6]

NOTE 1 The interpretation of "normal delivery" might be dependent on the service characteristics for the postal service appropriate to the individual postal item.

NOTE 2 The return address is usually (but not always) the postal address of the mail originator or the mail submitter. It need not necessarily be explicitly represented on the postal item – for example, it may be derived from a company logo or from a franking mark, or it may only be apparent when the postal item is opened (normally in a special location designated for the processing of non-deliverable postal items). It might also be impossible to determine the return address, in which case the non-delivered postal item concerned should be handled according to national regulations (e.g. be destroyed).

3.16.1.24
syntactically correct postal address
postal address in which the combination of postal address components is fully in accord with relevant standards and with relevant national or regional rules which define restrictions on allowed combinations and internal structures of such components

[UPU S42a-6]

NOTE 1 For example:

```
form_of_address: Mr.
given_name: John
surname: Smith
street_number_or_plot: 4395
thoroughfare_name: Station
thoroughfare_type: Road
town: Porchester
distribution_area_indicator: FAREHAM
postcode: PO16 8BQ
country: UNITED KINGDOM
```

forms a syntactically correct United Kingdom postal address, but if the country were France, it would not be syntactically correct, because France uses only numeric postcodes.

NOTE 2 Syntactic correctness does not imply validity. The above is not a valid postal address because the delivery point identified within it does not exist.

3.16.1.25
UPU
Universal Postal Union

3.16.1.26
valid postal address
postal address in which the combination of postal address components corresponds to, and provides for unambiguous identification of, a single delivery point and of an addressee and/or mailee

[UPU S42a-6]

NOTE 1 Valid postal addresses are not necessarily syntactically correct. For example:

```
function: The Secretary General
organization_name: CEN
street_number_or_plot: 36
thoroughfare_type: rue de
thoroughfare_name: Stassart
town: Bruxelles
country: BELGIUM
```

is not a syntactically correct postal address, because postcode is missing, but it is valid since there is only one rue de Stassart in Brussels (so it is unambiguous).

NOTE 2 The addressee and/or mailee specification may be implicit, as in the case in which the postal item is intended for the person(s) having legal access to the delivery point.

3.16.2 Postal address segments

3.16.2.1
addressee specification
postal address segment which specifies the addressee

[UPU S42a-6]

NOTE 1 The addressee specification is composed of either individual identification or organization identification, possibly combined with addressee role descriptor.

NOTE 2 Specification of the addressee may be optional or mandatory, depending on the particular postal service for which a postal address is to be used. For example, for normal letter mail, a delivery point specification is sufficient in many countries, and in this case, the addressee is considered as being any party which has legal access to the delivery point. In contrast, registered mail must normally carry an explicit specification of the addressee.

3.16.2.2
delivery point specification
postal address segment which designates the delivery point for a postal item

[UPU S42a-6]

NOTE 1 Delivery point specification is composed of defining authority, country, locality and delivery point location and/or postal delivery service point identifier. A postcode may also be required.

NOTE 2 The association between a delivery point specification and the delivery point may be service or time dependent. For example, a normal letter mail item addressed to an apartment may be delivered to a letter box in the entry hall of the apartment building; a registered mail item carrying an identical postal address must be delivered to the addressee (or his representative), possibly at the door of the apartment itself. Similarly, the link between a business reply or freepost service number and a delivery point may change if the customer concerned moves locations.

NOTE 3 Several delivery point specifications may be associated with a single delivery point.

NOTE 4 In some countries, certain forms of delivery point specification are limited to particular postal products. For example, a box number may not be appropriate for the addressing of recorded delivery postal items or parcels.

3.16.2.3
mail recipient dispatching information
postal address segment providing information intended for the routing and dispatch of mail by the mail recipient, when this is not the addressee

[UPU S42a-6]

NOTE 1 Mail recipient dispatching information is intended for use by the mailee, if one is specified, or by the mail recipient. It is not used by the postal operator.

NOTE 2 For postal items addressed to an organization and which are delivered by the postal operator to a mailroom or post office box, mail recipient dispatching information may include information such as wing, floor and door which, in the case of more specific services (such as registered mail) form part of the delivery point specification. Supplementary dispatch data may also be required.

3.16.2.4
mailee specification
postal address segment which specifies the mailee

[UPU S42a-6]

NOTE 1 Mailee specification is composed of individual identification or organization identification, possibly combined with mailee role descriptor.

NOTE 2 Specification of a mailee is required only in situations in which the postal operator is requested to deliver the postal item into the care of an individual or organization other than the addressee.

3.16.3 Postal address constructs

3.16.3.1
compound surname
postal address construct which identifies a family or provides indication of parentage

[UPU S42a-6]

NOTE 1 Compound surname is a component of individual identification. It comprises surname prefix and surname.

NOTE 2 The division of compound surname into two elements is intended for use where only part of the construct is significant for sorting purposes. If all words of a compound surname are significant for sorting purposes, surname prefix is not used.

NOTE 3 Patronymic and matronymic names, mother's maiden names, etc. are considered, for the purposes of this standard, as compound surnames. For example, in certain cultures, children's compound surnames are derived by appending "son" or "daughter" to either the father's first given name (patronymic names) or the mother's first given name (matronymic names); in others, a child's mother's maiden name and father's compound surname may be used in combination, though it may be that one of these may be considered as the person's preferred or legal compound surname.

NOTE 4 If an individual has more than one compound surname, these may be used separately or in combination. For example, in certain countries, a married person may be addressed either by their original compound surname, or by that of their spouse, or by a concatenation of the two. Word combinations that may appear only in combination should be regarded as a single compound surname.

NOTE 5 Where an individual has multiple compound surnames, the order may be significant.

3.16.3.2
country level information
postal address construct encompassing the postal address elements applying to countries or groupings of countries

[UPU S42a-6]

NOTE 1 Country level information is a component of delivery point specification. It comprises country name, country code, multi-country region, and international routing information.

NOTE 2 As a rule, these elements are only included in address presentation for cross border mail, but in that situation, they are necessary to avoid risk of ambiguity.

NOTE 3 The UPU strongly recommends that country level information be rendered in upper case wherever possible, be presented following all other address elements, and not presented on the same line with other address elements, while being expressed in the language of the sending country or in an internationally known language.

3.16.3.3
delivery point location
postal address construct identifying a delivery point, or a group of delivery points from which the postal operator may choose one, by reference to geographical and, where necessary, other spatial data expressed in human intelligible form

[UPU S42a-6]

NOTE 1 Delivery point location is a component of delivery point specification. It comprises thoroughfare, thoroughfare access data, street number or plot, delivery point access data and extension identifier.

NOTE 2 Delivery point location is relative to, and unique only within, country and locality.

NOTE 3 Delivery point access data may not be needed if the location of the delivery point on the plot is self evident. Thus, in a simple case, in which there is only one building, with one delivery point, on a plot, thoroughfare and street number or plot should be sufficient. If, as in the case of there being two residences on the plot, there are multiple delivery points, the combination of thoroughfare, street number or plot and extension identifier may be sufficient.

3.16.3.4
individual identification
postal address construct identifying, for the purpose of establishing the addressee or mailee of a postal item, either a single individual or a group of individuals, from which the postal operator may select one

[UPU S42a-6]

NOTE Individual identification is a component of addressee specification and mailee specification. It comprises form of address, given name, compound surname, name qualifier and qualification in which each element may occur none, one or more times.

3.16.3.5
locality
postal address construct identifying the geographical area in or adjacent to which a delivery point is located

[UPU S42a-6]

NOTE 1 Locality is a component of delivery point specification. It comprises region, proximate town, town, district/sector and delivery service qualifier.

NOTE 2 Region, town and district provide for multiple levels of geographically localizing information. Use need only be made of the number of levels which are actually required to unambiguously identify the geographic area in which the delivery point is situated. Thus:

- region should be used, in accordance with the specifications of the postal operator, if there are multiple towns having the same name within the country;

– though many towns are divided into commonly accepted areas or districts, the district/sector need not always be specified in a postal address if the address is otherwise unambiguous.

NOTE 3 Mobile delivery points, such as mobile homes and ships, may not be (permanently) situated in a particular country and locality. Nevertheless, they are associated with a country and locality for delivery point specification purposes. Depending on the situation, these might correspond either to the place of registration or to the place in which the delivery point is currently located or is expected to move.

3.16.3.6
organization identification
postal address construct identifying, for the purpose of establishing the addressee or mailee of a postal item, either a single individual or a group of individuals within an organization, from which the postal operator may select one

[UPU S42a-6]

NOTE 1 Organization identification is a component of addressee specification and mailee specification. It comprises function, organizational unit, organization name and legal status.

NOTE 2 Organization identification does not include the name of an individual which, if present, forms part of an individual identification. In a postal address which includes both an individual identification and an organization identification, one identifies the addressee of the postal item and the other identifies a mailee.

NOTE 3 Function and organizational unit are optional, the (group of) individual(s) then identified being the authorized representative(s) of the organization. Legal status may also be optional, if organization name is sufficient to unambiguously identify the intended organization.

3.16.3.7
service point identifier
postal address construct which designates a delivery point, or a group of delivery points from which the postal operator may choose one, by reference to a postal delivery service defined identifier, rather than by reference to its physical location

[UPU S42a-6]

NOTE 1 Service point identifier is a component of delivery point specification. It comprises delivery service type and delivery service indicator.

NOTE 2 Service point identifier is relative to, and unique only within, country and locality.

EXAMPLE post office box numbers, poste restante and business reply services.

3.16.4 Postal address elements

3.16.4.1
addressee role descriptor
postal address element indicating, in an addressee specification segment, that the role of the identified individual or organization is that of addressee

[UPU S42a-6]

NOTE 1 The purpose of addressee role descriptor is to ensure, when a postal address includes multiple addressee specifications or both an addressee specification and a mailee specification, that there is no ambiguity between them.

NOTE 2 Addressee role descriptor is optional. If it is omitted in cases in which the postal address contains both an addressee specification and a mailee specification, the distinction between the two segments has to be inferred from the mailee specification, from the order of the segments or from postal operator and product rules.

EXAMPLES

Attn.
tav (ter attentie van)
FAO
or (indicates that two addressees are considered as alternatives)
and (indicates that two addressees are considered as forming a group)

3.16.4.2
alternate delivery service identifier
postal address element which designates a delivery point, or a group of delivery points from which an alternate delivery service may choose one, by reference to a defined identifier, rather than by reference to its physical location

[UPU S42a-6]

NOTE Alternate delivery service identifier appears in the delivery point specification segment. It comprises element subtypes for alternate delivery service type and alternate delivery service indicator. An alternate delivery service type is an element sub-type indicating the type of delivery service. An alternate delivery service indicator is an element sub-type designating a specific delivery point, within the category identified by delivery service type, within, or accessed for delivery services via, the locality.

EXAMPLE private mail box.

3.16.4.3
building/construction
postal address element identifying the number or name and type of the building or construction in or adjacent to which a delivery point is located

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment. It comprises the element sub-types preceding building/construction type, succeeding building/construction type, and building/construction indicator.

EXAMPLES

Bâtiment
Block
Houseboat
Mobile Home

3.16.4.4
country code
postal address element designating the ISO 3166-1 code for the country, territory or area of geopolitical interest, in which a delivery point is located or via which the delivery point is accessed

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment.

NOTE 2 The ISO 3166-1 two character alphabetic representation is specified.

NOTE 3 In certain circumstances the country code may appear in an address presentation for cross border mail.

EXAMPLES

FR
NL
NZ

3.16.4.5
country name
postal address element designating the country, dependency or area of geopolitical interest, in which a delivery point is located or via which the delivery point is accessed

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment.

NOTE 2 In specifying the country name, the language used may be significant.

NOTE 3 Mobile delivery points, such as mobile homes and ships, might not be (permanently) located in or accessed via a particular country. Nevertheless, they are associated with a country and locality for delivery point specification purposes. Depending on the situation, these might correspond either to the place of registration or to the place in which the delivery point is currently located or is expected to move.

3.16.4.6
defining authority
postal address element designating the postal operator or other authority responsible for the definition and maintenance of the delivery point specification concerned

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment.

NOTE 2 Depending on the country, delivery point specifications may be defined and maintained by a central government agency, by regional or municipal authorities or by a postal operator.

NOTE 3 In a competitive postal service environment, a delivery point may be owned or served exclusively by a particular postal operator. In such a case, the defining authority for the delivery point specification will normally be the identity of the postal operator which owns or serves the delivery point concerned. Even where this is not the case, different operators may have different ways of specifying a particular delivery point. For example, in the U.K., Hays has its own system of "DX codes" which differ from the postcodes in use by The Post Office.

3.16.4.7
delivery service qualifier
postal address element designating the name of the distribution office used to for delivery services

[UPU S42a-6]

EXAMPLES
BORDEAUX CEDEX
NANTES CEDEX 1
FUTUROSCOPE CEDEX

3.16.4.8
delivery service identifier
postal address element which designates a delivery point, or a group of delivery points from which the postal operator may choose one, by reference to a defined identifier, rather than by reference to its physical location

[UPU S42a-6]

NOTE A postal delivery service identifier appears in the delivery point specification segment. It comprises element subtypes for delivery service type and delivery service indicator. A delivery service type is an element sub-type indicating the type of delivery service. A delivery service indicator is an element sub-type designating a specific delivery point, within the category identified by delivery service type, within, or accessed for postal delivery services via, the locality.

EXAMPLES post office box numbers, BP (Boîte Postale), PRIVATE BAG, poste restante and business reply services.

3.16.4.9
delivery service qualifier
postal address element designating the name of the distribution office used for delivery services

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment.

EXAMPLES post office box numbers, BP (Boîte Postale), PRIVATE BAG, poste restante and business reply services.

3.16.4.10
district/sector
postal address element giving the name of the hamlet, estate, or area within or adjacent to town, in which a delivery point is located, or via which it is accessed for postal delivery purposes

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment. It comprises element sub-types for four instances of district/sector and for a type and indicator for each instance.

NOTE 2 A district/sector may be a commonly known name for an area, or it may be an area assigned for a postal or administrative purposes. A district or sector may be one of a number of areas with a similar naming structure that may include a type and indicator structure.

EXAMPLES Arrondissement, Conjunto, Colonia Juarez, Kebele 4, Moo 11.

3.16.4.11
door
postal address element indicating the apartment, room or office in, at or adjacent to which a delivery point which is situated within a building is located

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment and in the mail recipient dispatching information segment. In each segment, it comprises the element sub-types door type and door indicator.

3.16.4.12
extension designation
postal address element designating the specific delivery point where this is not uniquely identified, within country and locality, by other components of delivery point location

[UPU S42a-6]

NOTE 1 For example, where all the delivery points for a block of apartments are located in the entry hall of a building, these may be distinguished by the allocation of a box number or by the use of the apartment number.

NOTE 2 For example, where all the delivery points for a block of apartments are located in the entry hall of a building, these may be distinguished by the allocation of a box number or by the use of the apartment number.

NOTE 3 Extension designation may not be required if there is only one delivery point on the plot, or in the vicinity defined by delivery point access data.

NOTE 4 In a country with multiple forms of secondary designator, these may be differentiated in a database or in address presentation, or they may be combined under a general description such as extension designation or door.

3.16.4.13
floor
postal address element indicating the floor or level on which a delivery point is located in a multi-story construction

[UPU S42a-6]

3.16.4.14
form of address
postal address element indicating, through a word, group of words, acronyms or abbreviations, an individual or group's civil status or condition

[UPU S42a-6]

NOTE 1 This element appears in the addressee specification segment and in the mailee specification segment.

NOTE 2 Form of address may include gender specific references and honorific distinctions, though preceding qualification is best suited for earned or designated attributes applying to an individual.

EXAMPLES 1
Mr.
Mrs.
Mr. & Mrs.
Miss
Family
Herr
Senora

NOTE 3 A form of address may in some countries be sufficient to identify an abstract addressee.

EXAMPLES 2
Postal Customer
Occupant
Current Resident

3.16.4.15
function
postal address element designating role or responsibility within an organization

[UPU S42a-6]

NOTE 1 This element appears in the addressee specification segment and in the mailee specification segment.

NOTE 2 Function, which relates to a role within an organization, should be distinguished from qualification, which is an intrinsic attribute of a specific individual.

NOTE 3 If there is a function, it implies that there is also an organisation even though an organisation might not be present in the address.

EXAMPLE 1 The function Postmaster may be followed by a town and postcode, omitting reference to the Post.

NOTE 4 An individual addressee may be denoted only by a function, for example because the name of the individual may not be known.

EXAMPLES 2
Managing Director
Chief Executive
Marketing Manager
Programmer
Janitor
Secretariat CEN/TC 331

3.16.4.16
given name
postal address element specifying the name used to distinguish between persons having the same compound surname(s) and who may have access to a particular delivery point

[UPU S42a-6]

NOTE 1 This element appears in the addressee specification segment and in the mailee specification segment. In each segment, it comprises element sub-types for given name part 1, given name part 2, and given name part 3. These can be used in rendition to shorten or eliminate parts of the given name while retaining other parts in full.

NOTE 2 If more than one given name is specified, the sequence of given names is significant. One may be defined as "first" or "preferred" given name.

NOTE 3 Given names may be abbreviated (e.g. Ch for Charles) or represented only by an initial letter.

NOTE 4 Given name is associated with an individual, as opposed to a family or a matrilineal or patrilineal identifier.

3.16.4.17
international routing information
postal address element indicating how a country, territory or area of geopolitical interest may be reached

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment.

EXAMPLE **VIA CAPE TOWN**

3.16.4.18
legal status
postal address element indicating the legal status of an organization

[UPU S42a-6]

EXAMPLES

GmbH
Inc.
Ltd.
AB
A/S
OY

3.16.4.19
mailee role descriptor
postal address element indicating, in association with an individual or organization identification, that the role of the identified individual or group is that of mailee

[UPU S42a-6]

NOTE 1 This element appears in the mailee specification segment.

NOTE 2 The purpose of mailee role descriptor is to ensure, when a postal address includes multiple mailee specifications or both an addressee specification and a mailee specification, that there is no ambiguity between them.

NOTE 3 Mailee role descriptor is optional. If it is omitted in cases in which the postal address contains both an addressee specification and a mailee specification, the distinction between the two segments has to be inferred from the addressee specification, from the order of the segments or from postal operator and product rules.

EXAMPLES

c/o (care of)
p/a (per adres)
or (indicates that two mailees are considered as alternatives)
and (indicates that two mailees are considered as forming a group)

3.16.4.20
multi-country region
postal address element indicating a region in which the country, territory, or area of geopolitical interest is located and by which it may be more effectively recognized

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment.

EXAMPLE **British West Indies (BWI)**.

3.16.4.21
name qualifier
postal address element used to distinguish between persons with the same compound surname(s) which have similar given names or initials

[UPU S42a-6]

NOTE This element appears in the addressee specification segment and the mailee specification segment.

EXAMPLES
III
Senior
the Third

3.16.4.22
organization name
postal address element giving the official name, the registered business name or other official designation of an organization

[UPU S42a-6]

NOTE This element appears in the addressee specification segment and in the mailee specification segment. In the mailee segment it comprises element sub-types for preceding organization name and succeeding organization name.

3.16.4.23
organization unit
postal address element identifying a subdivision of an organization

[UPU S42a-6]

NOTE This element appears in the addressee specification segment and in the mailee specification segment. In each segment it comprises element sub-types for two organizational levels. In the mailee segment it further comprises the element sub-types preceding organizational unit and succeeding organizational unit.

EXAMPLES
Marketing Department
Accounts Receivable

3.16.4.24
postcode
postal address element designating the code used for the sorting of mail

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment. It comprises the element sub-types primary postcode, secondary postcode and tertiary postcode.

NOTE 2 In many countries, postcodes are structured into two or more parts, with one part identifying the delivery region or postal processing facility at which delivery sorting should take place, the second defining the delivery office or route, within the area covered by that facility, and the third, if used, indicating the specific delivery point. For example,

most French postcodes commence with the 2-digit number of the Département; British ones are separated into two parts, with the first being a two, three or four character code which indicates the postal district and the second identifying a (group of) delivery addresses within this. Therefore, this element comprises the element sub-types primary postcode, secondary postcode and tertiary postcode.

NOTE 3 Postcodes may also be referred to as postal codes, ZIPs or ZIP-codes.

NOTE 4 Postcodes are not used in all countries. In many cases they are complementary information, providing only an encoded representation of locality, the (part of the) delivery route which includes the delivery point concerned and, possibly, the individual delivery point on that delivery route.

NOTE 5 A postcode can relate to a single delivery point or to a group of delivery points which are related in postal processing terms, usually by virtue of their being served by a single delivery office or being on a single delivery route. It may, however, relate to other grouping parameters, such as special services.

NOTE 6 Though normally having long-term, national significance, postcodes can be operator specific (c.f. Hays DX codes in the United Kingdom) and might have only temporary existence, as when a special postcode is assigned to handle mail resulting from a charity appeal, or when an existing assignment of codes is reformed due to changes in the scope or magnitude of delivery point distribution.

NOTE 7 Though defined primarily for the purpose of sorting mail, postcodes are often used, outside the postal processing context, for other purposes. In particular, many organizations use them in marketing databases to link potential customer characteristics to geographic areas.

3.16.4.25
qualification
postal address element indicating an individual's professional or academic qualification or rank in a professional group or society

[UPU S42a-6]

NOTE 1 This element comprises the element sub-types preceding qualification, intermediate qualification and succeeding qualification.

NOTE 2 Qualification, which is an attribute of an individual, should be distinguished from function, which designates a role within an organization. An individual's qualification(s) remain valid, irrespective of changes in the organization for which (s)he works or in his or her function or job title in an organization.

EXAMPLES

Reverend
PhD
Fellow of the Royal Society
FRS
Barrister at Law

3.16.4.26
region
postal address element specifying the geographic or administrative area of the country in which town is situated

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment. It comprises element sub-types for three instances of region and for a type and indicator for each instance.

NOTE 2 Regions are generally related to administrative rather than to postal geography. Examples include French Departments, German Länder, British Counties and American States. See also ISO 3166–2 Country Subdivision Code.

NOTE 3 Region as a postal address element may become less significant over time if the combination of other elements such as town and postcode is unique within the country. It may still be included in the address to corroborate other information, though there may be other more efficient ways to do that.

3.16.4.27
stairwell
postal address element indicating access to floor or door within a building/construction

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment and in the mail recipient dispatching information segment. In each segment, it comprises the element sub-types stairwell type and stairwell indicator.

EXAMPLE **Escalier**

3.16.4.28
street number or plot
postal address element designating the area, or the object on an area, adjacent to thoroughfare, in which the delivery point or delivery point access is located

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment. It is comprised of element sub-types for type and indicator.

NOTE 2 This may be in the form of a house or site number or name and will normally correspond to an area defined in the cadastral or municipal register of building plots.

NOTE 3 Where one building/construction spans several registered plots, this element may be composite, e.g. 6–8. This situation is hard to distinguish from the use of an extension designation following the street number or plot, or from appending a secondary identifier, such as door, to the street number or plot. Generally, local or country knowledge will allow understanding of which use is indicated.

3.16.4.29
supplementary delivery point data
postal address element providing additional data or instructions intended to facilitate access to, or designation of, a delivery point

[UPU S42a-6]

EXAMPLES
Opposite number 23
50 metres to the left of the main door

3.16.4.30
supplementary dispatch data
postal address element providing additional data or instructions intended to assist the mail recipient in the processing of a postal item

[UPU S42a-6]

NOTE This element appears in the mail recipient dispatching information segment.

EXAMPLES An internal organizational mail distribution code, or mail stop.

3.16.4.31
surname
postal address element consisting of the root or part of a compound surname which has sorting significance

[UPU S42a-6]

NOTE 1 This element appears in the addressee specification segment and in the mailee specification segment. In each segment, it comprises sub-types for surname part 1 and surname part 2. These can be used to index names that are not sorted on the part of the surname that is rendered first, or optionally may be rendered first.

NOTE 2 For countries in which surnames are rendered before given names in a consistent manner, template ordering can reflect this situation. If the rendering within a country is not consistent, an implementation can support multiple orderings provided that it has a mechanism for signaling which ordering is preferred in a given instance.

3.16.4.32
surname prefix
postal address element consisting of the prefix or part of a compound surname which is not significant for sorting purposes

[UPU S42a-6]

NOTE This element appears in the addressee specification segment and in the mailee specification segment.

EXAMPLES
de
van
van de
von

3.16.4.33
thoroughfare
postal address construct which identifies the road or part of a road or other access route along which a delivery point may be accessed

[UPU S42a-6]

NOTE 1 This element appears in the delivery point specification segment. It comprises sub-types for three occurrences, specified as primary, secondary and tertiary. Within each occurrence there are sub-types for name, name prefix, type and qualifier, with the latter two further sub-typed as preceding or succeeding.

NOTE 2 For addressing purposes, a thoroughfare need not be on land, e.g. a canal or river might serve as a thoroughfare in the address of a houseboat or of a construction on the bank.

NOTE 3 A thoroughfare name may uniquely identify the thoroughfare or may need to be supplemented with type and qualifier information or other elements in order to be unique in the required context.

EXAMPLES 1
San marcos
Pine Ridge
Main
6th
Charles de Gaulle

NOTE 4 A thoroughfare name prefix may be used to separate connecting words without sorting significance from the main part of the name of the thoroughfare.

EXAMPLES 2
"de la" in Avenue de la Republique
"of the" in Avenue of the Americas

NOTE 5 A thoroughfare type indicates the category or type of thoroughfare. Thoroughfare type can be used to distinguish between instances in the locality which have the same thoroughfare name. Thoroughfare type is separated from thoroughfare name and thoroughfare qualifier because it may have different abbreviation rules and/or a sorting significance which differs from its relative position in printed representations.

NOTE 6 Thoroughfare type may precede or follow thoroughfare name in printed representations; its position may depend on national, regional and/or linguistic considerations, or may be specific to the thoroughfare concerned. For example, in Belgium, French language thoroughfare types, such as boulevard and drêve du generally precede the thoroughfare name, whilst their Flemish equivalents, laan and dreef, follow the thoroughfare name.

EXAMPLES 3

Avenue
Beach
Canal
Lane
Place
Road
Square
Street

NOTE 7 A thoroughfare qualifier distinguishes between different parts or instances of thoroughfare, within a locality, which have the same thoroughfare name and thoroughfare type.

NOTE 8 Thoroughfare qualifier may be separated from thoroughfare name if it has different abbreviation rules and/or has a position in printed representations which is not adjacent to thoroughfare name or thoroughfare type. Its position in printed representations — at the beginning, between thoroughfare name and thoroughfare type, or at the end — may be determined by national, regional and/or linguistic considerations, or may be specific to the thoroughfare concerned.

EXAMPLES 4

Directionals such as **North**, **SW**
Qualifiers such as **Little**, **Upper**

NOTE 9 A secondary thoroughfare identifies the road or part of a road or other thoroughfare in which a delivery point may be reached and which is accessed via primary thoroughfare.

NOTE 10 A tertiary thoroughfare identifies the road or part of a road or other thoroughfare in which a delivery point may be reached and which is accessed via a primary thoroughfare and secondary thoroughfare.

3.16.4.34
town
postal address element indicating the name of the village, town or city in which a delivery point is located, or near to or via which the delivery point is accessed for postal delivery purposes

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment.

3.16.4.35
wing
postal address element identifying, for a delivery point, the building/construction section in which it is housed and/or the main entry door through which it is accessed

[UPU S42a-6]

NOTE This element appears in the delivery point specification segment and in the mail recipient dispatching information segment. In each segment, it comprises the element sub-types wing type and wing indicator.

3.16.5 Postal address element sub-types

3.16.5.1
indicator
term used in the names of postal address element sub-types, representing a logical part of a root element, which may be combined with a type to constitute an identifier, and instances of which represent numerical, alphabetic, or symbolic data that differentiates one instance of an element from another, within a certain scope of reference

[UPU S42a-6]

EXAMPLE In **Apartment A**, **Apartment** is a type and **A** is an indicator. **Apartment A** should be a unique identifier within a limited scope of reference.

3.16.5.2
instance
term used in the names of postal address element sub-types, which represents an occurrence of the root element

[UPU S42a-6]

EXAMPLE In Brazil, **Quadra 7** may be the name of an instance of district/sector.

3.16.5.3
intermediate
term used in the names of postal address element sub-types, which represents a position after preceding and before succeeding

[UPU S42a-6]

EXAMPLE In **Prof. Alex graaf van Nispen BA MKM**, **graaf** is an intermediate qualification.

3.16.5.4
level
term used in the names of postal address element sub-types, which represents an unspecified hierarchical ordering

[UPU S42a-6]

EXAMPLE
organisational unit level 1

3.16.5.5
name
term used in the name of postal address element sub-types, which represents a logical part of a root element, and may be further differentiated by the content of related element sub-types

[UPU S42a-6]

NOTE The term name is also used in the name of address elements, such as organization name and given name. Thoroughfare name is an element sub-type and not an element in its own right because it is subordinate to primary, secondary and tertiary thoroughfare, which are element sub-types.

EXAMPLE Thoroughfare name is further differentiated by the content of thoroughfare name and thoroughfare type.

3.16.5.6
part
term used in the names of postal address element sub-types, which represents a physical subdivision of the root element, such as a word or delimited string

[UPU S42a-6]

NOTE 1 Physical and logical parts are both differentiated using the second digit of the element sub-type code, but the names of logical parts use other terms defined in this section, such as type, indicator and qualifier.

NOTE 2 The given name Jean Claude may be stored as two physical parts, or it may be stored in the root element. If the name is hyphenated, such as Jean-Claude, it could be stored as two physical parts only if the presence of the hyphen is managed by some convention governing retention or restoration during rendition, and in view of those complexities, it is likely to be stored as a single physical part.

EXAMPLES For physical parts:
given name part 2
surname part 2

3.16.5.7
position
term used in the names of postal address element sub-types, which represents an instance of the root element that can be combined with other instances either within a single address or in a set of addresses to be processed within a particular template

[UPU S42a-6]

EXAMPLES
Supplementary delivery point data position 1

3.16.5.8
preceding
term used in the names of postal address element sub-types, which represents a position before intermediate and before succeeding

[UPU S42a-6]

EXAMPLE In **Prof. Alex graaf van Nispen BA MKM**, **Prof.** is a preceding qualification.

3.16.5.9
prefix
term used in the names of postal address element sub-types, which represents a position before another element sub-type

[UPU S42a-6]

NOTE The term prefix is also used in the name of address elements, and in that case represents a position before another element. For example, a surname prefix comes before the surname. Thoroughfare name prefix is an element sub-type and not an element in its own right because it is subordinate to primary, secondary and tertiary thoroughfare, which are element subtypes.

3.16.5.10
primary
term used in the names of postal address element sub-types, which represents a status or level above secondary and above tertiary

[UPU S42a-6]

NOTE Primary, secondary and tertiary may be used for both instances and parts.

EXAMPLES
primary thoroughfare
primary postcode

3.16.5.11
qualifier
term used in the name of postal address element sub-types, which represents a logical part of a root element, and further differentiates the content of related element sub-types

[UPU S42a-6]

NOTE The term qualifier is also used in the name of address elements, and in that case differentiates the content of related elements. For example, name qualifier further differentiates the content of given name and surname. Thoroughfare qualifier is an element sub-type and not an element in its own right because it is subordinate to primary, secondary and tertiary thoroughfare, which are element sub-types.

EXAMPLE Thoroughfare qualifier further differentiates the content of thoroughfare name and thoroughfare type.

3.16.5.12
secondary
term used in the names of postal address element sub-types, which represents a status or level below primary and above tertiary

[UPU S42a-6]

NOTE Primary, secondary and tertiary may be used for both instances and parts.

EXAMPLES
secondary thoroughfare
secondary postcode

3.16.5.13
succeeding
term used in the names of postal address element sub-types, which represents a position after preceding and after intermediate

[UPU S42a-6]

EXAMPLE In **Prof. Alex graaf van Nispen BA MKM**, **MKM** is a succeeding qualification.

3.16.5.14
tertiary
term used in the names of postal address element sub-types, which represents a status or level below primary and below secondary

[UPU S42a-6]

NOTE Primary, secondary and tertiary may be used for both instances and parts.

EXAMPLES
tertiary thoroughfare
tertiary postcode

3.16.5.15
type
term used in the names of postal address element sub-types, representing a logical part of a root element, which may be combined with an indicator to constitute an identifier, and instances of which describe a category

[UPU S42a-6]

EXAMPLE In **RESIDENCE MASUREL, RESIDENCE** is a type.

3.16.6 Other terms and definitions

3.16.6.1
unrendered address
postal address independent of formatting

EXAMPLE Presenting a postal address as a series of postal address segments according to UPU S42a-6 is an unrendered address.

3.17 Module 17-specific terminology: Data structure for ITU-T E.164 phone number data

3.17.1
extension
telephone number within a PBX

EXAMPLES
0012125551212 (a telephone number that conforms to the numbering plan in several European countries)
0-22-749-0111 (a local telephone number within Switzerland)

3.17.2
international numbering plan
telephone numbers that conform to ITU-T E.164

EXAMPLES
+12125551212
+41-22-749-0111 (punctuation and spaces are ignored)

3.17.3
phone number element
portion of a phone number

EXAMPLE The phone number **+12125551212+234** may be decomposed into the elements **+** (international numbering plan prefix), **1** (country code), **212** (NPA), **555** (NNX), **1212** (line), **234** (extension). The phone number **+41227490111** may be decomposed into the elements **+** (international numbering plan prefix), **41** (country code), **22** (city code), **7490111** (local number).

3.17.4
PBX
private branch exchange
telephony equipment and services for private use

NOTE Typically, the telephones in a PBX each have their own numbers — these numbers are called *extensions*.

3.17.5
private numbering plan
telephone numbers that are permitted not to conform to ITU-T E.164

EXAMPLES
0012125551212 (a telephone number that conforms to the numbering plan in several European countries)
0-22-749-0111 (a local telephone number within Switzerland)

3.18 Module 18-specific terminology: Data structure for who-what-where-when-why-how (W5H) event data

There are no additional definitions for Module 18.

3.19 Module 19-specific terminology: Data structure for entity-person-group (EPG) contact data

There are no additional definitions for Module 19.

3.20 Module 20-specific terminology: Data structure for entity-person-group (EPG) security credentials data

3.20.1
authentication
act of verifying the claimed identity of an entity

[ISO/IEC 2382-8:1998]

3.20.2
access control
means of ensuring that the resources of a data processing system can be accessed only by authorized entities in authorized ways

[ISO/IEC 2382-8:1998]

3.21 Module 21-specific terminology: Data structure for entity-person-group (EPG) relationships and grouping data

3.21.1
entity-person-group
EPG
entity or a person or a group of entities and/or persons

4 Structure of this International Standard

This International Standard is organized in three portions:

— **Subclauses 3.10 and onward:** Module-specific terminology.

— **Clauses 10 and onward:** Module-specific provisions, such as the definitions of the data structures, datatypes, and other interoperability requirements.

— **Everything else:** Wording that is common to all metadata modules.

The structure of each module Clause (Clauses 10 and onward) is organized as the following subclauses:

— **Subclause x.1, Introduction to Module:** Specific information or commentary about the technical content of the Clause, its purpose, and/or the reasons prompting its preparation.

— **Subclause x.2, Scope of Module:** A scoping statement for the module, similar to a Scope clause of a standard.

— **Subclause x.3, Functional Capabilities:** The intended capabilities and applications of the module.

— **Subclause x.4, Abstract Model:** An abstraction of the data structure(s), presented as object-modeling using UML, including descriptions of classes, attributes, and relations. Although class modeling is used, the classes can be considered “plain old data structures” and no object-oriented implementation is required.

— **Subclause x.5, Computational Description and Datatypes:** Maps and renders the object-model into datatypes, components, and subcomponents. The ISO/IEC 11404 datatype notation is used, which is binding-independent. While the terms “class” and “datatype” refer to different degrees of implementation specificity, each individual class shares the same concepts with its corresponding datatype.

— **Subclause x.6, Additional Provisions for Bindings:** Some modules might have additional provisions for bindings beyond those in Clause 5.

— **Subclause x.7, Additional Provisions for Conformity:** Some modules might have additional provisions for conformity, conformance, and conformance labeling beyond those in Clause 6.

Within each module Clause (Clauses 10 and onward), the following provisions apply:

— Datatypes are defined using ISO/IEC 11404 notation.

— Provisions embedded in 11404 comments (signaled by “`//`”) are normative.

— The provision “all components optional” written as an 11404 comment means that (normatively) the obligation attribute is "optional" for all data elements contained within the structure.

— The abbreviation SPM is used, which means “smallest permissible maximum”. The SPM value is intended to give implementers a lower limit on conforming implementations. Applications should not assume that implementations support capabilities beyond the SPM value unless prior arrangements have been made. For example, the data declaration “**`characterstring, // SPM: 8192 characters`**” means that the datatype is a **`characterstring`** has a smallest permitted maximum of 8192 characters, i.e., all implementations of this datatype that must be able to store at least 8192 characters.

— The provision “all sizes are SPM” written as an 11404 comment means that (normatively) the size values are smallest permitted maximum values.

5 Bindings

Bindings should conform to the ISO/IEC 20944 interoperability bindings.[2)]

6 Conformance

ISO/IEC 20944-1, Clause 4, Conformance, is incorporated by reference.

Conformity may be claimed to all of 19773 or to individual clauses of 19773.

7 Designation of internationally standardized items

7.1 Designation suffix syntax

Designations for internationally standardized items are specified in ISO/IEC Directives, Part 2 (2011), Annex G.

The standards designation pattern **`ISO/IEC 19773`** with the suffix **`/C/xxxxx`** is equivalent to the statement “ISO/IEC 19773, Clause xxxxx”. The use of a hyphen (-) indicates a range of clauses, e.g., **`ISO/IEC 19773/C/10-14`** refers to Clauses 10 through 14, inclusive. The use of a comma (,) indicate non-contiguous values, e.g., **`ISO/IEC 19773/C/10-12,14`** refers to Clauses 10 through 14, excluding Clause 13.

EXAMPLE A product or service labeled with “Conforms to ISO/IEC 19773/C/19” means that the product or service conforms Clause 19 on EPG Contact Data (which implies conformity to other Clauses).

7.2 Designation suffixes for profiles

The designation pattern **`ISO/IEC 19773`** with the suffix **`/P/xxxxx`** is equivalent to the statement “ISO/IEC 19773, profile xxxxx” where profile designation is defined in Clause 8. Hyphens may be used to indicate ranges and commas may be used to indicate non-contiguous values, as described in 7.1.

2) Implementers of this International Standard are encouraged (but not required) to make use of ISO/IEC 20944 interoperability bindings.

8 Profile designations

The following are standard profile[3] designations for this International Standard.[4]

— "16-21090": Module 16, postal data, corresponds to the profile of ISO 21090, i.e., the use of an ISO 21090 implementation within an ISO/IEC 19773 module 16 implementation. (See subclause 16.7.)

9 Clause reserved for future use

Clause reserved for future use.

10 Module 10: Data structure for reference-or-literal (reflit)

10.1 Introduction to module

Module 10 is the reference/literal data structure that can be used for storing data (i.e., a literal value) or a reference to data (e.g., the value stored is a pointer to the real data). This approach provides a run-time choice of literal vs. reference, in contrast to compile-time choice (e.g., making the choice of reference or literal when a program or query is written) or a specification-time choice (e.g., choosing within a technical specification, such as a standard, whether the choice of reference or literal is required).

10.2 Scope of module

This Clause provides the description of the data structure for holding a reference to data, or a literal data value — to be chosen at run-time.

10.3 Functional capabilities

The **reflit** data structure allows a unit of data (a value) to be expressed directly (via a literal value) or expressed indirectly (via a reference to a value), and for the choice of reference or literal to be made at run-time.

EXAMPLE 1 By using a **reflit** data structure, a description[5] may be stored directly (e.g., characterstring literal) or indirectly (e.g., URL to document). The following is an example uses **`characterstring`** as the base datatype[6]:

```
description : reflit(characterstring) // a reference to or literal of a characterstring
```

The following code stores the string "a large ball" as a (characterstring) literal, i.e., **`description`** contains a copy of the string:

```
description.reflit_kind = literal;
description.literal_value_as_text = "a large ball";
```

3) These profile designations describe subsets, supersets, and/or configurations of this International Standard, as per ISO/IEC TR 10000-1.

4) For example, the suffix **`/P/16-21090`** corresponds to the profile described in the first bullet item of Clause 8.

5) For the purposes of presentation, the same illustration of a description (represented as a characterstring) is used throughout the examples in this Clause. Other examples are possible, such as aggregate data structures of non-textual data which may be stored directly (literal) or indirectly (reference).

6) The **reflit** data structure may use any datatype as a base datatype.

The following code stores the string “a large ball” as a reference (to a string), i.e., **description** contains a pointer to the file that contains the string:

```
description.reflit_kind = reference;
description.reference_value_as_text = "http://website.com/ball_description.txt";
```

The **reflit** data structure enables choosing at run-time whether the value is available directly or indirectly. A run-time decision may be desirable when the choice of literal data or referenced data is not known in advance of usage or where making a choice, in advance, make result in impractical implementation, usage, or storage or data.

EXAMPLE 2 It may be practical to store small descriptions (e.g., a paragraph) in a database, but it may be impractical to store large descriptions (e.g., a whole book or movie) in a database. By using a **reflit** data structure, the small strings may be stored literally and the large strings may be stored via reference.

EXAMPLE 3 Regardless of the size of the description, it might be appropriate to have a singular copy of the description so that only one string needs to be changed when the description changes, i.e., changing all the “a large ball” strings to “a large red ball” is affected by changing the contents of the file **"ball_description.txt"**. In this case, a reference would be used.

```
description_1.reflit_kind = reference;
description_1.reference_value_as_text = "http://website.com/ball_description.txt";
description_2.reflit_kind = reference;
description_2.reference_value_as_text = "http://website.com/ball_description.txt";
```

EXAMPLE 4 Regardless of the size of the description, it might be appropriate to have individual copies of each of the descriptions so that they can be updated independently. In this case, a literal would be used.

10.4 Abstract model

10.4.1 General

Conceptually, a **reflit** is the union of a pointer-to-a-datatype (reference) and the instance-of-a-datatype (literal). Additional data is required to identify the kind of reference value or literal value, its encoding (concerns data representation, not datatype), and its presentation as a text stream (an array of characters) or as a binary stream (an array of octets).[7)] The remainder of this subclause is an object-model description[8)] of a **reflit**. This object model is mapped to binding-independent semantics in subclause 10.6.

7) The choice between text and binary representation is based upon which optimization strategy best suits the intended needs. The text represention offers the most portability across system implementations, word sizes, byte orderings, and hardware data alignment requirements; however, a text representation can require significant computation when certain datatypes are transformed from their native representation to a text form (e.g., floating point numbers). The binary representation offers the best space efficiency and computational efficiency; however, the binary representation might differ across various machine architectures and operating systems, and the binary representation requires choosing a text encoding for characterstring datatypes (which will not be optimal for all reportiores of characters).

8) In this International Standard, the classes can be considered “plain old data structures”. Clause 10.4 presents the object-model, which is discussed in terms of classes, attributes, and relations. Subclause 10.5 renders maps the object-model into datatypes, components, and subcomponents. While the terms “class” and “datatype” refer to different concepts, in this International Standard each individual class shares the same concepts with its corresponding datatype.

10.4.2 reflit(of_type)

The **reflit** is the class that is instantiated to create a reference-or-literal. The class is parameterized in that it takes **of_type** as a parameter to the class instantiation. For example, a `reflit(characterstring)` is a data structure that can hold a `characterstring` datatype (e.g., `"hello world"`) or a pointer to that string (e.g., `"http://hello.com/world.txt"`); a `reflit(space_extent)` is a data structure that can hold a `space_extent` datatype (e.g., `"30N35W-40N45W"`) or a pointer to that space-time region (`&lat_lon_range`). This class contains the following components:

— **reflit_kind : state(reference, literal, none):** The **reflit_kind** may have the values: **reference** (means the **reflit** holds a reference), **literal** (means the **reflit** holds a literal), and **none** (means the **reflit** does not hold data, e.g., an empty **reflit** object).[9)]

— **reference_value : reference_type(of_type):** The reference itself, as a value. The reference may be dereferenced for indirect access to the value.

— **literal_value : literal_type(of_type):** The literal value itself.

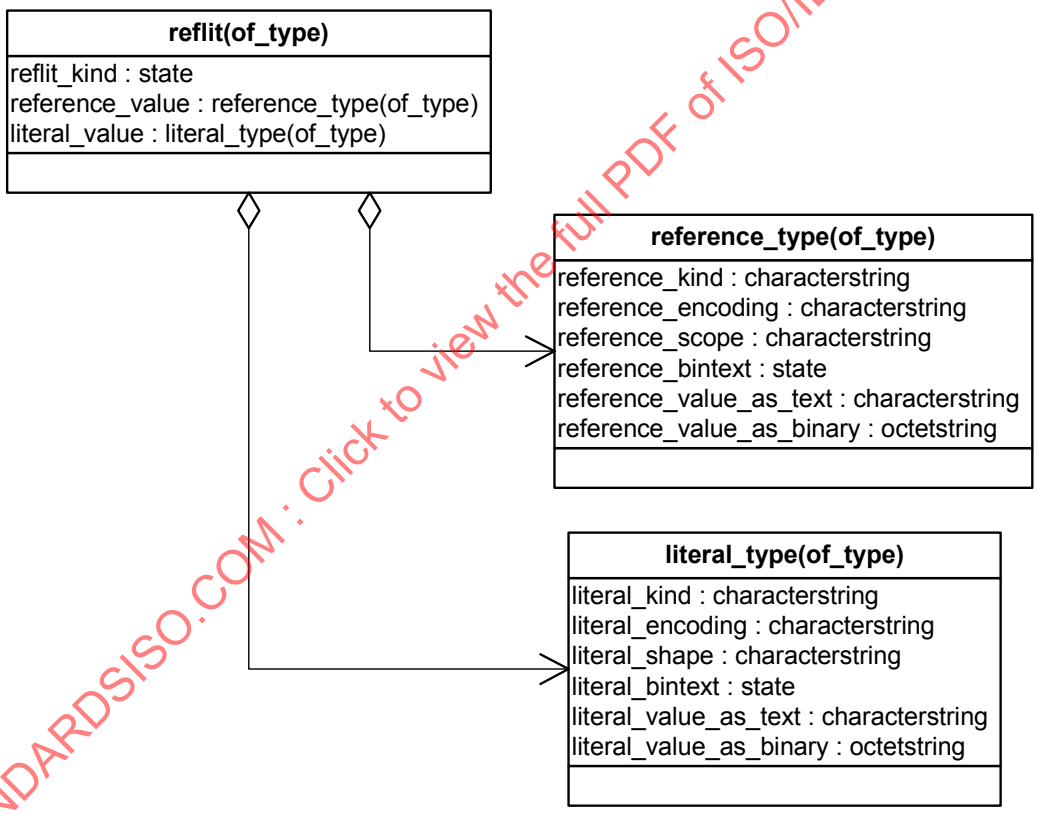

Figure 1 — UML presentation of: reflit, reference_type, literal_type

9) Because a reflit cannot simultaneously hold both a reference and a literal, it is possible that implementations use common storage for both, i.e., there is no requirement for persistence of data when changing from a literal to a reference (and vice versa), and storage should be cleared when changing from one to the other (which might be done automatically for certain bindings).

EXAMPLE The following are examples of navigable data objects in the above UML diagram:

self.reflit_kind
self.reference_value
self.reference_value.reference_kind
self.reference_value.reference_encoding
self.reference_value.reference_scope
self.reference_value.reference_bintext
self.reference_value.reference_value_as_text
self.reference_value.reference_value_as_binary
self.literal_value
self.literal_value.literal_kind
self.literal_value.literal_encoding
self.literal_value.literal_bintext
self.literal_value.literal_shape
self.literal_value.literal_value_as_text
self.literal_value.literal_value_as_binary

10.4.3 reference_type(of_type)

The **reference_type** class holds a reference to data. The class is parameterized in that it takes **of_type** as a parameter to the class instantiation. This class contains the following components.

— **reference_kind : characterstring:** The kind of reference, e.g., **`"uri"`**. The component **reference_kind** describes the functional aspects of the reference and, possibly, not the representational aspects of the reference. This data element is permitted to have the following values:

  — `"uri"`: A Uniform Resource Identifier, as specified by RFC 3986.

  — `"iri"`: An Internationalized Resource Identifier, as specified by RFC 3987.

  — `"pointer"`: An implementation-defined pointer, as specified by ISO/IEC 9899.

  — `"filename"`: A filename (also known as a pathname), as specified by ISO/IEC 9945.[10)]

  — `"other"`: A reference specified in a format described outside this International Standard.

  — `"x-*"`: An implementation-defined format in accordance with IANA procedures.

— **reference_encoding : characterstring:** The representation technique, e.g., **`"far32/uint64"`** (a far-32 pointer as an unsigned 64-bit integer), **`"near16/hex"`** (a near-16 pointer as a hexadecimal number), **`"ascii"`** (a text encoding in ASCII). The suffix "LE" is little-endian ordering and “BE” is big-endian ordering; no suffix means the byte ordering is unspecified. This data element is permitted to have the following values:

  — `( "near", "_", ( "8" | "16" | "32" | "40" | "48" | "56" | "64" | "72" | "96" | "128" | "256" ), ( "" | "LE" | "BE" ) )`: A near pointer, i.e., a pointer of flat address space. The number that follows `"near"` is the number of bits of precision in the pointer, e.g., `"near_32"` is a 32-bit pointer of a flat addressing space.

  — `( "far", { "_", ( "8" | "16" | "32" | "40" | "48" | "64" ), ( "" | "LE" | "BE" ) } )`: A far pointer, i.e., a pointer of segmented address space. The numbers that follows `"far"` is the number of bits of the segments of the pointer with the last segment as a flat address space, e.g., `"far_16_32"` is a 48-bit pointer of 2^{16} possible segments of 32-bit flat address spaces.

  — `( "pic", "_", ( "8" | "12" | "16" | "24" | "32" | "40" | "48" | "64" ), ( "" | "LE" | "BE" ) )`: A position-independent code (PIC) pointer. A PIC pointer is relative to its place of use. The number that follows `"pic"` is the number of bits of precision in the pointer, e.g., `"pic_16"` is a 16-bit relative pointer.

10) The token “filename” is defined as a “pathname”, as per ISO/IEC 9945.

— ("uint", "_", ("8" | "16" | "24" | "32" | "48" | "64" | "128" | "256" | "512" | "1024"), ("" | "LE" | "BE")): A twos complement unsigned integer. The number that follows “uint” is the number of bits of precision in the integer, e.g., “uint_16” is a 16-bit unsigned integer with a range of 0 to 2^{16}-1 (65535) inclusive.

— `( "int", "_", ( "8" | "16" | "24" | "32" | "48" | "64" | "128" | "256" | "512" | "1024" ), ( "" | "LE" | "BE" ) )`: A twos complement signed integer. The number that follows `"int"` is the number of bits of precision in the integer, e.g., `"int_16"` is a 16-bit signed integer with a range of -2^{15} (–32768) to $+2^{15}$–1 (+32767) inclusive.

— `( "tcdui", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ), ( "" | "LE" | "BE" ) } )`: A text-coded unsigned decimal integer consisting of one or more digits 0 through 9, represented as ISO 646 characters `"0"` through `"9"`. The number that follows `"tcdui"` is the number of digits of precision in the integer, e.g., `"tcdui_4"` is a 4-digit integer with a range of 0000 to 9999 inclusive.

— `( "bcdui", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ) }, ( "" | "LE" | "BE" ) )`: A binary-coded decimal integer that uses binary values 0000 through 1001 for the decimal digits 0 through 9. The number that follows `"bcdui"` is the number of digits of precision in the integer, e.g., `"bcdui_4"` is a 4-digit integer with a range of 0000 to 9999 inclusive.

— `( "tchi", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ) }, ( "" | "LE" | "BE" ) )`: A text-coded hexdecimal integer that uses ISO 646 characters `"0"` through `"9"` and `"A"` through `"F"` to represent the hexadecimal digits. The number that follows `"tchi"` is the number of digits of precision in the hexadecimal integer, e.g., `"tchi_4"` is a 4-digit hexadecimal integer with a range of 0000 to FFFF inclusive.

— `"pointer"`: An implementation-defined pointer, as specified by ISO/IEC 9899.

— `"filename"`: A filename, as specified by ISO/IEC 9945.

— `"other"`: A reference specified in a format described outside this International Standard.

— `"x-*"`: An implementation-defined format.

— **reference_scope : characterstring:** The scope (e.g., boundaries) in which the reference is understood, e.g., **`"proc/1234"`** (a pointer within the address space of process 1234), **`"stackframe/0x27463720"`** (a scope-linkage structure in a programming language), **`"namespace.org/object_id_list"`** (a namespace label that provides a namespace of identifiers). The structure and format of this component is outside the scope of this Clause

— `( "stackframe/", hex_digit_string )`: The reference is within an execution stack frame, as specified by the hex_digit_string.

— `( "proc/", characterstring )`: The reference is within a process address space where the characterstring is the process identifier.

— `( "xmldom/", characterstring )`: The reference is within the XML document object model frame as specified by the characterstring.

— `( "other/", characterstring )`: The reference is within the scope defined by the characterstring, which is an implementation-defined scope.

— **reference_bintext : state(binary, text, none):** The **reference_bintext** may have the values: **binary** (means **reference_value_as_binary** holds the reference value), **text** (means **reference_value_as_text** holds the reference value), and **none** (means no value is held, e.g., an empty **reference_type** object).

— **reference_value_as_text : characterstring:** The representation of the reference value as a string of characters.

— **reference_value_as_binary : octetstring:** The representation of the reference value as a string of octets.

10.4.4 literal_type(of_type)

The **literal_type** class holds a literal data value. The class is parameterized in that it takes **of_type** as a parameter to the class instantiation. This class contains the following components.

— **literal_kind : characterstring:** The kind of literal, i.e., its datatype. The component **literal_kind** describes the functional aspects of the literal value and, possibly, not the representational aspects of the literal value. This characterstring is in the format **`"datatype // datatype_source"`**, for example **`"int // Java"`** would correspond to an **`int`** datatype within the Java programming language.

— **literal_encoding : characterstring:** The representation technique, e.g., **`"ascii"`** (encoded in ASCII), **`"utf8"`** (encodied in ISO/IEC 10646 UTF-8). This data element is permitted to have the following values:

  — `( "near", "_", ( "8" | "16" | "32" | "40" | "48" | "56" | "64" | "72" | "96" | "128" | "256" ) )`: A near pointer, i.e., a pointer of flat address space. The number that follows `"near"` is the number of bits of precision in the pointer, e.g., `"near_32"` is a 32-bit pointer of a flat addressing space.

  — `( "far", { "_", ( "8" | "16" | "32" | "40" | "48" | "64" ) } )`: A far pointer, i.e., a pointer of segmented address space. The numbers that follows `"far"` is the number of bits of the segments of the pointer with the last segment as a flat address space, e.g., `"far_16_32"` is a 48-bit pointer of 2^{16} possible segments of 32-bit flat address spaces.

  — `( "pic", "_", ( "8" | "12" | "16" | "24" | "32" | "40" | "48" | "64" ) )`: A position-independent code (PIC) pointer. A PIC pointer is relative to its place of use. The number that follows `"pic"` is the number of bits of precision in the pointer, e.g., `"pic_16"` is a 16-bit relative pointer.

  — `( "uint", "_", ( "8" | "16" | "24" | "32" | "48" | "64" | "128" | "256" | "512" | "1024" ) )`: A twos complement unsigned integer. The number that follows `"uint"` is the number of bits of precision in the integer, e.g., `"uint_16"` is a 16-bit unsigned integer with a range of 0 to 2^{16}-1 (65535) inclusive.

  — `( "int", "_", ( "8" | "16" | "24" | "32" | "48" | "64" | "128" | "256" | "512" | "1024" ) )`: A twos complement signed integer. The number that follows `"int"` is the number of bits of precision in the integer, e.g., `"int_16"` is a 16-bit signed integer with a range of -2^{15} (–32768) to $+2^{15}$–1 (+32767) inclusive.

  — `( "tcdui", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ) } )`: A text-coded unsigned decimal integer consisting of one or more digits 0 through 9, represented as ISO 646 characters `"0"` through `"9"`. The number that follows `"tcdui"` is the number of digits of precision in the integer, e.g., `"tcdui_4"` is a 4-digit integer with a range of 0000 to 9999 inclusive.

  — `( "bcdui", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ) } )`: A binary-coded decimal integer that uses binary values 0000 through 1001 for the decimal digits 0 through 9. The number that follows `"bcdui"` is the number of digits of precision in the integer, e.g., `"bcdui_4"` is a 4-digit integer with a range of 0000 to 9999 inclusive.

  — `( "tchi", "_", { ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ) } )`: A text-coded hexdecimal integer that uses ISO 646 characters `"0"` through `"9"` and `"A"` through `"F"` to represent the hexadecimal digits. The number that follows `"tchi"` is the number of digits of precision in the hexadecimal integer, e.g., `"tchi_4"` is a 4-digit hexadecimal integer with a range of 0000 to FFFF inclusive.

— (("le" | "be" | "un"), "_", ("0" | "1" | "4" | "8" | "16" | "24" | "32" | "48" | "64" | "96" | "128" | "256" | "512" | "1204"): An octetstring in little endian (LE), big endian (BE), or unspecified (UN) ordering of a defined bit size, specified by the number after the underscore.

— `"pointer"`: An implementation-defined pointer, as specified by ISO/IEC 9899.

— `"filename"`: A filename, as specified by ISO/IEC 9945.

— `"other"`: A reference specified in a format described outside this International Standard.

— `"x-*"`: An implementation-defined format.

— **literal_shape : characterstring:** The run-time sizing of the literal data, e.g., a **`characterstring`** datatype has no size that is inherent to the datatype definitions, but all instantiations of a **`characterstring`** datatype have a run-time size. This characterstring is in one of the three formats: (1) the null string (**`""`**) that represents the shape of a scalar, (2) *integer_value*, a string of decimal digitals that represents the length of the array, e.g., **`"7"`** indicates a vector of elements of length 7, (3) **`"(integer1 integer2 ... integerN)"`**, a string of space-separated decimal values surrounded by parentheses that are the dimension of a multi-dimensional array, e.g., **`"(7 2 3)"`** indicates a 7×2×3 multidimensional array.

— **literal_bintext : state(binary, text, none):** The **literal_bintext** may have the values: **binary** (means **literal_value_as_binary** holds the literal value), **text** (means **literal_value_as_text** holds the literal value), and **none** (means no value is held, e.g., an empty **literal_type** object).

— **literal_value_as_text : characterstring:** The representation of the literal value as a string of characters, e.g., "hello world".

— **literal_value_as_binary : octetstring:** The representation of the literal value as a string of octets, e.g., { 68 65 6C 6C 6F 20 77 6F 72 6C 64 }[11].

10.5 Computational description and datatypes

10.5.1 General

The **reflit** datatype is a fundamental datatype and it is not dependent upon any other modules.

10.5.2 reflit(of_type)

ISO/IEC 11404 definition

```
type reflit(of_type) = record
(
    reflit_kind:
        state(reference, literal, none),
    reference_value:
        reference_type(of_type),
    literal_value:
        literal_type(of_type),
),
```

Description

See 10.4.2 for a description of the record and its components.

11) The string of octets spells "hello world" encoded in ISO 646.

10.5.3 reference_type(of_type)

ISO/IEC 11404 definition

```
type reference_type(of_type) = record
(
    reference_kind:
        characterstring,    // SPM: 8192 characters
    reference_encoding:
        characterstring,    // SPM: 8192 characters
    reference_scope:
        characterstring,    // SPM: 8192 characters
    reference_bintext:
        state(binary, text, none),
    reference_value_as_text:
        characterstring,  // Shall be large enough to hold "pointer(of_type)".
    reference_value_as_binary:
        octetstring,  // Shall be large enough to hold "pointer(of_type)".
),
```

Description

See 10.4.3 for a description of the record and its components.

10.5.4 literal_type(of_type)

ISO/IEC 11404 definition

```
type literal_type(of_type) = record
(
    literal_kind:
        characterstring,    // SPM: 8192 characters
    literal_encoding:
        characterstring,    // SPM: 8192 characters
    literal_shape:
        characterstring,    // SPM: 8192 characters
    literal_bintext:
        state(binary, text, none),
    literal_value_as_text:
        characterstring,  // Shall be large enough to hold "of_type".
    literal_value_as_binary:
        octetstring,  // Shall be large enough to hold "of_type".
),
```

Description

See 10.4.4 for a description of the record and its components.

10.6 Additional provisions for bindings

The scope of a reference (i.e., the kinds of objects it can point to) is implementation-defined.

The visibility of a reference (i.e., where it can be used) is implementation-defined.

The lifetime of a reference (i.e., the duration in which the reference remains valid) is implementation-defined.

10.7 Additional provisions for conformity

There are no additional provisions for conformity.

11 Module 11: Data structure for multiple internationalized/localized values and data

11.1 Introduction to module

Clause 11 is the multivalue: a set of values that may have different presentation, but present the same meaning. The multivalue may be used for applications that need several presentations of, say, a concept, yet the presentations are different based upon the context of use. The context of use may describe variations, such as differences in: country, region, culture, language, mode of communication (e.g., visual), method of communication (e.g., sign language), human functioning, environment, etc.. The multidata is similar to a multivalue except that a multidata can have different datatypes for each presentation whereas the multivalue uses the same base datatype.

11.2 Scope of module

This Clause provides the description of the data structure for holding several values whose meaning is the same, yet their presentation may differ.

NOTE Presentations may differ due to differences in: country, region, culture, language, mode of communication (e.g., visual), method of communication (e.g., sign language), human functioning, environment, etc..

11.3 Functional capabilities

11.3.1 General

This subclause describes intended capabilities and applications of this Clause.

11.3.2 The multivalue data structure

The **multivalue** data structure stores multiple different presentations of the same concept[12)], each presentation associated with a particular context of use.

NOTE A **multivalue** data structure is similar to the **multistring** data structure defined in Clause 12 with the following difference: the **multistring** is defined for the base datatype **`characterstring`** whereas the **multivalue** is parameterized by a user-defined base datatype.

EXAMPLE For the datatype **`dance_performance_venue`** defined as:

```
type dance_performance_venue = record
(
    location: characterstring, // location of event
    timing: characterstring, // timing of event
),

example_instance : dance_performance_venue = ( "New York", "2000-01-01" ),
```

12) A single concept may have several equivalent presentations that all the same meaning, e.g., presentations in different languages.

The multivalue based upon the datatype **dance_performance_venue** can be used to store equivalent British English and US Spanish presentations:

```
type multivalue_dance_performance_venue = multivalue(dance_performance_venue),  // creates the
new datatype

event_in_new_york_city : multivalue_dance_performance_venue =  // instantiates the new datatype
(
      contextualized_value_list =
      (
        ////////////////////////////////////////////////////////////////////
        (   // British English presentation
            context_identifier = 'language="en-GB"',
            value = ( where = "New York", when = "17.03.2002" ),
        ),
        ////////////////////////////////////////////////////////////////////
        (   // US Spanish presentation
            context_identifier = 'language="es-US"',
            value = ( where = "Nueva York", when = "03/17/2002" ),
        ),
        ////////////////////////////////////////////////////////////////////
      ),
),
```

11.3.3 The multidata data structure

The **multidata** data structure stores multiple different presentations of the same concept, each presentation associated with a particular context of use *and* permits the individual presentations to have different datatypes.

NOTE A **multidata** data structure is similar to the **multivalue** data structure with the following difference: the **multivalue** has the same user-defined base datatype for each presentation within an instance of the **multivalue** data structure, whereas the **multidata** has a user-defined datatype, possibly different, for each of its presentations.

EXAMPLE The following example shows using the multidata data structure for storing several presentations of the concept "the number 2":

```
concept_of_number_2 : multidata =  // instantiates the multidata data structure
(
      contextualized_value_list =
      (
        ////////////////////////////////////////////////////////////////////
        (   // English text presentation (literal value stored)
            context_identifier = 'language="en",mime-content-type=text/plain',
            data = ( reflit_kind = literal, literal_value =
                    (
                        literal_kind = "text",
                        literal_encoding = "ascii",
                        literal_bintext = text,
                        literal_value_as_text = "two",
                    ) ),
        ),
        ////////////////////////////////////////////////////////////////////
        (   // German HTML text presentation (reference to URL)
            context_identifier = 'language="de",mime-content-type=text/html',
            data = ( reflit_kind = reference, reference_value =
                    (
                        reference_kind = "url",
                        reference_bintext = text,
                        reference_value_as_text = "http://zwei.org/zwei.html",
                    ) ),
```

```
            ),
            ////////////////////////////////////////////////////////////////////
            (   // graphical presentation of numeral 2 (reference to GIF image)
                context_identifier = 'mime-content-type=image/gif,
                data = ( reflit_kind = reference, reference_value =
                        (
                            reference_kind = "url",
                            reference_bintext = text,
                            reference_value_as_text = "http://zwei.org/2.gif",
                        ) ),
            ),
            ////////////////////////////////////////////////////////////////////
            (   // German audio presentation of numeral 2 (reference to WAV file)
                context_identifier = 'mime-content-type=audio/wav,
                data = ( reflit_kind = reference, reference_value =
                        (
                            reference_kind = "url",
                            reference_bintext = text,
                            reference_value_as_text = "http://zwei.org/zwei.wav",
                        ) ),
            ),
            ////////////////////////////////////////////////////////////////////
            (   // Morse Code presentation of numeral 2 (literal value stored)
                context_identifier = 'mime-content-type=other/x-morse-code',
                data = ( reflit_kind = literal, literal_value =
                        (
                            literal_kind = "text",
                            literal_encoding = "ascii",
                            literal_bintext = text,
                            literal_value_as_text = "..---",
                        ) ),
            ),
            ////////////////////////////////////////////////////////////////////
        ),
    ),
```

11.4 Abstract model

11.4.1 General

This subclause is an object-model description of the **multivalue** data structure and the **multidata** data structure. This object model is mapped to binding-independent semantics in subclause 11.6.

11.4.2 multivalue

11.4.2.1 General

The following is a UML presentation of the **multivalue** and **contextualized_value** classes:

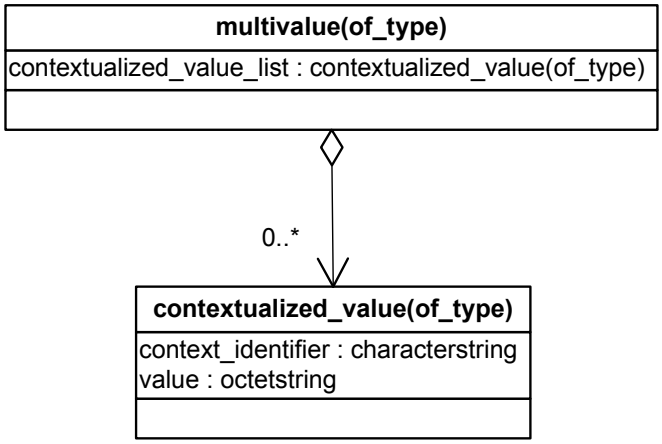

Figure 2 — UML presentation of: multivalue, contextualized_value

11.4.2.2 multivalue(of_type)

The **multivalue** class contains the following components:

— **contextualized_value_list : array (0..*) of contextualized_value(of_type):** The array of values all having the same meaning, each in their own context.

11.4.2.3 contextualized_value

The **contextualized_value** class contains the following components:

— **context_identifier : characterstring:** The identifier that is associated with the context, such as a description of the context.

— **value : of_type:** The data value represented in the **of_type** datatype[13].

EXAMPLE The following are examples of navigable data objects in the above UML diagram:

```
self.contextualized_value_list
self.contextualized_value_list[0].context_identifier
self.contextualized_value_list[0].value
self.contextualized_value_list[1].context_identifier
self.contextualized_value_list[1].value
```


13) The **of_type** datatype is the parameter to the **multivalue()** datatype.

11.4.3 multidata

11.4.3.1 General

The following is a UML presentation of the **multidata** and **contextualized_data** classes:

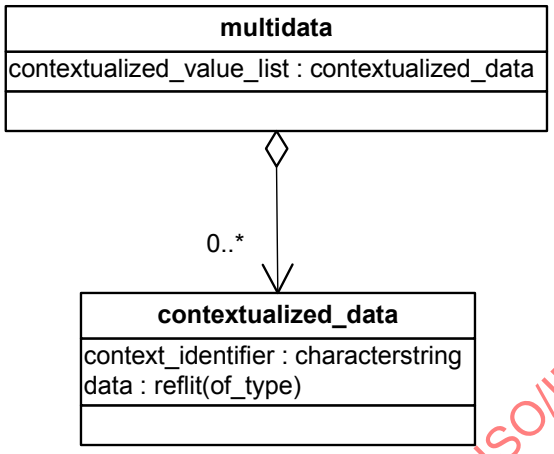

Figure 3 — UML presentation of: multidata, contextualized_data

11.4.3.2 multidata

The **multidata** class contains the following components:

— **contextualized_value_list : array (0..*) of contextualized_data:** The array of values all having the same meaning, each in their own context.

11.4.3.3 contextualized_data

The **contextualized_data** class contains the following components:

— **context_identifier : characterstring:** The identifier that is associated with the context, such as a description of the context.

— **data : reflit(of_type):** The individual presentation associated with the context identifier. The **reflit** data structure is defined in Clause 10.

EXAMPLE The following are examples of navigable data objects in the above UML diagram:

```
self.contextualized_value_list
self.contextualized_value_list[0].context_identifier
self.contextualized_value_list[0].data    // See footnote14)
self.contextualized_value_list[1].context_identifier
self.contextualized_value_list[1].data
```

14) The **data** component is a **reflit** data structure, which defined in Clause 12. Clause 12 provides additional navigation requirements such that the data component itself may be further navigated, e.g.:

```
self.contextualized_value_list[0].data.reflit_kind
self.contextualized_value_list[0].data.literal_value
self.contextualized_value_list[0].data.literal_value.literal_value_as_text
```

11.5 Computational description and datatypes

11.5.1 General

The **reflit** datatype of Clause 10 is used in the following definitions.

11.5.2 multivalue

11.5.2.1 General

These subclauses define the datatypes for the **multivalue** data structure.

11.5.2.2 multivalue(of_type)

ISO/IEC 11404 definition

```
type multivalue(of_type) = record
(
    contextualized_value_list:
        array (0..*) of contextualized_value_type(of_type),
),
```

Description

See 11.4.2.2 for a description of the record and its components.

11.5.2.3 contextualized_value_type(of_type)

ISO/IEC 11404 definition

```
type contextualized_value_type(of_type) = record
(
    context_identifier:
        characterstring,    // SPM: 4096 characters
    value:
        of_type,
),
```

Description

See 11.4.2.3 for a description of the record and its components.

11.5.3 multidata

11.5.3.1 General

These subclauses define the datatypes for the **multidata** data structure.

11.5.3.2 multidata

ISO/IEC 11404 definition

```
type multidata = record
(
    contextualized_value_list:
        array (0..*) of contextualized_data_type,
),
```

Description

See 11.4.3.2 for a description of the record and its components.

11.5.3.3 contextualized_data_type

ISO/IEC 11404 definition

```
type contextualized_data_type = record
(
    context_identifier:
        characterstring,    // SPM: 4096 characters
    data:
        reflit(unspecified),
),
```

Description

See 11.4.2.3 for a description of the record and its components.

11.6 Additional provisions for bindings

There are no additional provisions for bindings.

11.7 Additional provisions for conformity

There are no additional provisions for conformity.

12 Module 12: Data structure for multiple internationalized/localized strings and texts

12.1 Introduction to module

This Clause specifies the multistring and the multitext: a set of textual strings that may have different presentation, but present the same meaning. The multistrings might be used for applications that need several presentations of, say, a concept, yet the presentations are different based upon the context of use. The context of use may describe variations, such as differences in: country, region, culture, language, mode of communication (e.g., visual), method of communication (e.g., sign language), human functioning, environment, etc.. The multitext is similar to multistring except that the multistring uses a base datatype of a characterstring for all its presentations, while the multitext supports the use of different datatypes (e.g., JPEG image, WAV audio file) for the presentation of textual strings.

12.2 Scope of module

This Clause provides the description of the data structure for holding several textual strings whose meanings are the same, yet their presentation may differ.

12.3 Functional capabilities

12.3.1 General

The **reflit** datatype of Clause 10 is used in the definitions below.

12.3.2 The multistring data structure

The **multistring** data structure permits a textual string to have different presentations according to context of use.

NOTE A **multistring** data structure is similar to the **multivalue** data structure defined in Clause 11 with the following difference: the **multistring** is defined for the base datatype **characterstring** whereas the **multivalue** is parameterized by a user-defined base datatype.

EXAMPLE According to ISO/IEC Guide 2, subclause 1.1, the concept of "standardization" has equivalences in several languages. These different presentations can be stored in a multistring:

```
iso_iec_guide_2_standardization : multistring =  // instantiates the multistring
(
      contextualized_value_list =
      (     // strings in several languages all meaning "standardization"
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="en"', // English text string
            value_encoding = "ascii", value_as_text = "standardization", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="fr-CA"', // French Canadian text string
            value_encoding = "iso-8859-1", value_as_text = "normalisation", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="ru"', // Russian text string
            value_encoding = "iso-10646-1", value_as_text = "стандартизация", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="de"', // German text string
            value_encoding = "iso-8859-1", value_as_text = "Normung", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="es"', // Spanish text string
            value_encoding = "iso-8859-1", value_as_text = "normalización", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="it"', // Italian text string
            value_encoding = "ascii", value_as_text = "normazione", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="nl"', // Dutch text string
            value_encoding = "ascii", value_as_text = "nornalisatie", ),
        ////////////////////////////////////////////////////////////////////
        ( context_identifier = 'language="sv"', // Swedish text string
            value_encoding = "ascii", value_as_text = "standardisering", ),
        ////////////////////////////////////////////////////////////////////
      ),
),
```

12.3.3 The multitext data structure

The **multitext** data structure permits a textual string to have different presentations according to context of use, *and* permits the individual presentations to have different presentation forms.[15)]

NOTE A **multitext** data structure is similar to the **multistring** data structure with the following difference: the **multistring** uses textual strings, presented as a characterstring, as the base datatype for each presentation within an instance of the **multistring** data structure, whereas the **multitext** has a user-defined form, datatype, and execution-time-determined reference-or-literal[16)] for each of its presentations.

EXAMPLE The following example shows using the multidata data structure for storing several presentations of the concept "the number 2":

```
concept_of_number_2 : multitext =  // instantiates the multitext data structure
(
      contextualized_value_list =
```

15) For example, representation forms can be: plain text (e.g., ASCII text), images (e.g., JPEG), audio (e.g., WAV), etc..

16) The **reflit** (reference-or-literal) data structure is defined in Clause 10.

```
    (
      ////////////////////////////////////////////////////////////////////
      (   // English text presentation (literal value stored)
          context_identifier = 'language="en",mime-content-type=text/plain',
          data = ( reflit_kind = literal, literal_value =
                  (
                      literal_kind = "text",
                      literal_encoding = "ascii",
                      literal_bintext = text,
                      literal_value_as_text = "two",
                  ) ),
      ),
      ////////////////////////////////////////////////////////////////////
      (   // German HTML text presentation (reference to URL)
          context_identifier = 'language="de",mime-content-type=text/html',
          data = ( reflit_kind = reference, reference_value =
                  (
                      reference_kind = "url",
                      reference_bintext = text,
                      reference_value_as_text = "http://zwei.org/zwei.html",
                  ) ),
      ),
      ////////////////////////////////////////////////////////////////////
      (   // graphical presentation of numeral 2 (reference to GIF image)
          context_identifier = 'mime-content-type=image/gif,
          data = ( reflit_kind = reference, reference_value =
                  (
                      reference_kind = "url",
                      reference_bintext = text,
                      reference_value_as_text = "http://zwei.org/2.gif",
                  ) ),
      ),
      ////////////////////////////////////////////////////////////////////
      (   // German audio presentation of numeral 2 (reference to WAV file)
          context_identifier = 'mime-content-type=audio/wav,
          data = ( reflit_kind = reference, reference_value =
                  (
                      reference_kind = "url",
                      reference_bintext = text,
                      reference_value_as_text = "http://zwei.org/zwei.wav",
                  ) ),
      ),
      ////////////////////////////////////////////////////////////////////
      (   // Morse Code presentation of numeral 2 (literal value stored)
          context_identifier = 'mime-content-type=other/x-morse-code',
          data = ( reflit_kind = literal, literal_value =
                  (
                      literal_kind = "text",
                      literal_encoding = "ascii",
                      literal_bintext = text,
                      literal_value_as_text = "..---",
                  ) ),
      ),
      ////////////////////////////////////////////////////////////////////
    ),
),
```

12.4 Abstract model

12.4.1 General

This subclause is an object-model description of the **multistring** data structure and the **multitext** data structure. This object model is mapped to binding-independent semantics in subclause 12.6.

12.4.2 multistring

12.4.2.1 General

The following is a UML presentation of the **multistring** and **contextualized_string** classes:

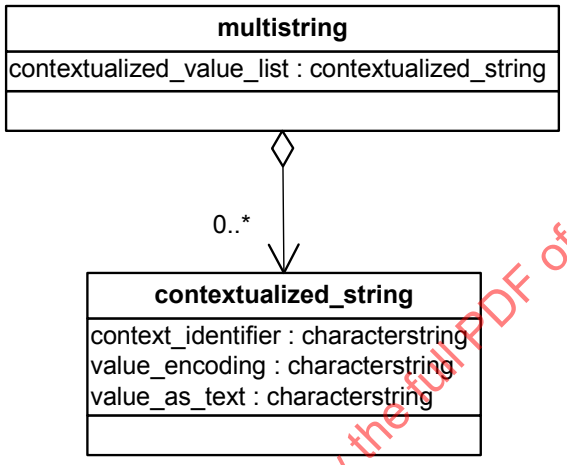

Figure 4 — UML presentation of: multistring, contextualized_string

12.4.2.2 multistring

The **multistring** class contains the following components:

— **contextualized_value_list : array (0..*) of contextualized_string:** The array of values all having the same meaning, each in their own context.

12.4.2.3 contextualized_string

The **contextualized_string** class contains the following components:

— **context_identifier : characterstring:** The identifier that is associated with the context, such as a description of the context. This data element is a string with zero or more of the following space-separated components:

  — `"language='langcode'"`: A language identifier, as defined by IETF RFC 5646.

  — `"mode='interactionmode'"`: A human interaction mode.

  — `"assertrequirements='specid'"`: Assertion of requirements, as per specification reference specid.

  — `"certifyrequirements='specid;cert'"`: Certification of requirements, as per specification reference specid as certified by cert.

  - "assertconformity='specid'": Assertion of requirements, as per specification reference specid.
  - "certifyconformity='specid'": Certification of conformity, as per specification reference specid as certified by cert.
  - "*": Format specified external to this Clause.[17)]

- **value_encoding : characterstring:** The string encoding method. This data element is permitted to have the following values:
  - "isoiec646": A string encoded in ISO/IEC 46.
  - "isoiec8859-1": A string encoded in ISO/IEC 8859-1.
  - "isoiec10646/utf-8": A string encoded in ISO/IEC 10646 UTF-8 encoding.
  - "isoiec10646/ucs-2": A string encoded in ISO/IEC 10646 UCS-2 encoding.
  - "isoiec10646/utf-16": A string encoded in ISO/IEC 10646 UTF-16 encoding with undefined octet ordering.
  - "isoiec10646/utf-16BE": A string encoded in ISO/IEC 10646 UTF-16 encoding with big endian octet ordering.
  - "isoiec10646/utf-16LE": A string encoded in ISO/IEC 10646 UTF-16 encoding with little endian octet ordering.
  - "ascii": A string encoded in ANSI X3.4.
- **value_as_text : characterstring:** The string for an individual presentation.

EXAMPLE The following are examples of navigable data objects in the above UML diagram:

self.contextualized_value_list
self.contextualized_value_list[0].context_identifier
self.contextualized_value_list[0].value_encoding
self.contextualized_value_list[0].value
self.contextualized_value_list[1].context_identifier
self.contextualized_value_list[1].value_encoding
self.contextualized_value_list[1].value

17) For example, module 17 further specifies the meaning of context_identifier within the use of module 17.

12.4.3 multitext

12.4.3.1 General

The following is a UML presentation of the **multidata** and **contextualized_data** classes:

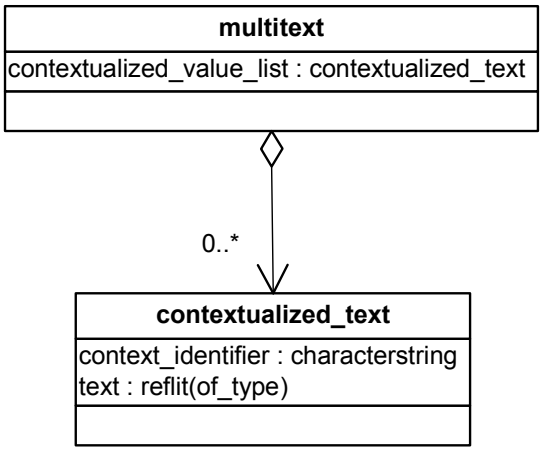

Figure 5 — UML presentation of: multitext, contextualized_text

12.4.3.2 multitext

The **multitext** class contains the following components:

— **contextualized_value_list : array (0..*) of contextualized_text:** The array of values all having the same meaning, each in their own context.

12.4.3.3 contextualized_text

The **contextualized_text** class contains the following components:

— **context_identifier : characterstring:** The identifier that is associated with the context, such as a description of the context.

— **text : reflit(unspecified):** The individual presentation associated with the context identifier. The **reflit** data structure is defined in Clause 10.

EXAMPLE The following are examples of navigable data objects in the above UML diagram:

self.contextualized_value_list
self.contextualized_value_list[0].context_identifier
self.contextualized_value_list[0].text // See footnote[18)]
self.contextualized_value_list[1].context_identifier
self.contextualized_value_list[1].text

18) The **text** component is a **reflit** data structure, which is defined in Clause 10. Clause 10 provides additional navigation requirements such that the data component itself may be further navigated, e.g.:

self.contextualized_value_list[0].text.reflit_kind
self.contextualized_value_list[0].text.literal_value
self.contextualized_value_list[0].texf.literal_value.literal_value_as_text

12.5 Computational description and datatypes

12.5.1 General

This subclause defines datatypes using ISO/IEC 11404 notation.

12.5.2 multistring

12.5.2.1 General

These subclauses define the datatypes for the **multistring** data structure.

12.5.2.2 multistring

ISO/IEC 11404 definition

```
type multistring = record
(
    contextualized_value_list:
        array (0..*) of contextualized_string_type,
),
```

Description

See 12.4.2.2 for a description of the record and its components.

12.5.2.3 contextualized_string_type

ISO/IEC 11404 definition

```
type contextualized_string_type = record
(
    context_identifier:
        characterstring,  // SPM: 4096 characters
    value_encoding:
        characterstring,  // SPM: 4096 characters
    value_as_text:
        characterstring,
),
```

Description

See 12.4.2.3 for a description of the record and its components.

12.5.3 multitext

12.5.3.1 General

These subclauses define the datatypes for the **multitext** data structure.

12.5.3.2 multitext

ISO/IEC 11404 definition

```
type multitext = record
(
    contextualized_value_list:
```

```
        array (0..*) of contextualized_text_type,
),
```

Description

See 12.4.3.2 for a description of the record and its components.

12.5.3.3 contextualized_text_type

ISO/IEC 11404 definition

```
type contextualized_text_type = record
(
    context_identifier:
        characterstring,    // SPM: 4096 characters
    text:
        reflit(unspecified),
),
```

Description

See 12.4.2.3 for a description of the record and its components.

12.6 Additional provisions for bindings

There are no additional provisions for bindings.

12.7 Additional provisions for conformity

There are no additional provisions for conformity.

13 Module 13: Data structure for slot tuple

13.1 Introduction to module

Clause 13 is the data structure the provides the structure of slot tuples, a mechanism for describing individual data values via the 3-tuple of { **identifier** , **kind** , **value** }. Clause 14 provides the data structure for bundling a table of unstructured (or structured) data values.

13.2 Scope of module

This Clause describes the data structure for slot tuples, a mechanism for describing individual data values.

NOTE Clause 14 provides the data structure for bundling a table of unstructured (or structured) data values.

13.3 Functional capabilities

The slot tuple data structure provides a unit of data storage, i.e., a data element whose identifier and kind are not pre-defined. The slot tuple consists of the following components: its identifier (e.g., label, identifier, name, etc.), its kind (e.g., datatype, taxon, etc.), and its value (a unit of computation, such as a string, record, array, set, date-time, boolean, integer, or real value).

NOTE The example in Clause 14 provides a table of slot tuples, which are used to represent a portion of the 11179-3 metamodel and institution-specific extensions (additional descriptive data outside the 11179-3 standard).

EXAMPLE The following examples are three slot tuples (using the text-text-text variant of the datatype), each representing a component of an E-mail message:

```
e_mail_from_line : slot_tuple_as_ttt,
e_mail_from_line.identifier      = "From",
e_mail_from_line.kind             = "ietf/rfc822.from/string",
e_mail_from_line.value            = "president@company.org",

e_mail_subject_line : slot_tuple_as_ttt,
e_mail_subject_line.identifier   = "Subject",
e_mail_subject_line.kind          = "ietf/rfc822.subject/string",
e_mail_subject_line.value         = "Letter from the Company President",

e_mail_date_line : slot_tuple_as_ttt,
e_mail_date_line.identifier      = "Date",
e_mail_date_line.kind             = "ietf/rfc822.date/date",
e_mail_date_line.value            = "2004-10-10T01:23:45.67",
```

The slot tuple may be used in several compatible datatypes, each optimized for a particular kind of usage:

13.4 Abstract model

13.4.1 General

Conceptually, a slot tuple is 3-tuple that represents a identifier, a kind, and a value. Typically, slot tuples are used in groups (see Clause 14) and used for general data interchange where datatypes might not be known prior to data interchange. The remainder of this subclause is an object-model description of an slot tuple. This object model is mapped to binding-independent semantics in subclause 13.5.

13.4.2 slot_tuple components

The **slot_tuple** class models the tuple of { **identifier**, **kind**, **value** }. The **slot_tuple** instances may use one of several implementation variants that are optimized for a particular use. The class contains the following components:

— **identifier : multidata:** A label associated with the **value** component of this **slot_tuple** instance. The conventions for choosing a label are outside the scope of this International Standard.[19] The following are examples of **identifier**:

  — `"latitude"`: A **slot_tuple**, whose **value** corresponds to a latitude of a latitude-longitude pair of values.[20] (This **identifier** could be stored in **multidata** or **characterstring** implementation variants of **slot_tuple**.)

  — `"17"` or `17`: The **identifier** 17 (as a string or a numeral) is used as an index (a kind of label) that corresponds to the 17th element in an array of **slot_tuple**, e.g., the 17th data point in a time series dataset. (The corresponding **value** components are the time series data themselves.)

  — `0x13 0xFC 0x3D 0x8A`: A **slot_tuple** whose **identifier** is an internal pointer or key that is associated with its corresponding **value** component. (This **identifier** could be stored in **multidata** or **octetstring** implementation variants of **slot_tuple**.)

19) The **identifier** must be consistent with the **slot_tuple** implementation variant chosen, e.g., if **slot_tuple_ttt** variant is chosen, then the **identifier** must be suitable for storing into a **characterstring** datatype, if **slot_tuple_bbb** variant is chosen, then the **identifier** must be suitable for storing into an **octetstring** datatype.

20) If an unstructured table of slot tuples is used (see Clause 14), then "latitude" **slot_tuple** might be stored adjacently to the another **slot_tuple** whose identifier is "longitude".

— **kind : multidata:** A classification taxon associated with the **value** component of this tuple instance. The conventions for choosing a **kind** are outside the scope of this International Standard.[21] The following are examples of **kind**:

  — `"iso_iec_11404.time(second,10,0)"`: A timestamp with precision to one second.

  — `"iso_iec_11404.scaled(10,2)"`: A scaled integer with two decimal digits (e.g., a datatype for currency in dollars and cents).

  — `"java.String"`: A Java string class.

  — `"uml.Integer"`: An integer datatype, as specified in UML.

  — `"binary"`: A kind of value in binary, possibly in contrast to a value in `"text"`.

— **value : multidata:** The **value** itself, as labeled by the **identifier** component and as typed by the **kind** component.[22] The following are examples of **value**:

  — `"20020317T223000-0500"`: A date-time corresponding to March 17, 10:30 PM, Eastern Standard Time.

  — `314.15`: A value corresponding to €314.15.

  — `"стандартизация"`: A Java String class instance for the word "standard" in Russian.

  — `2147483647`: A UML integer datatype corresponding to 2^{31}-1.

  — `0x00 0x01 0x02 0x03 0x04`: A 5-octet binary value.

13.4.3 slot_tuple and variants

The slot tuple may be used in several compatible datatypes, each optimized for a particular kind of usage:[23]

21) The **kind** must be consistent with the **slot_tuple** implementation variant chosen, e.g., if **slot_tuple_ttt** variant is chosen, then the **kind** must be suitable for storing into a **characterstring** datatype, if **slot_tuple_bbb** variant is chosen, then the **kind** must be suitable for storing into an **octetstring** datatype.

22) The **kind** component and, possibly in conjunction with the **identifier** component, should provide enough guidance for the required computations of the **value** within a defined context of use.

23) The **`reflit`** datatype is defined in Clause 10. The **`multidata`** datatype is defined in Clause 12. The **`characterstring`** and **`octetstring`** datatypes are defined in ISO/IEC 11404.

Slot variant	Datatype of Slot Component			Notes
	identifier	kind	value	
slot_tuple	multidata	multidata	multidata	The most general of the datatypes. All other slot variants may be promoted to this datatype.
slot_tuple_as_ttt	characterstring	characterstring	characterstring	For text-only applications.[24]
slot_tuple_as_ttrl	characterstring	characterstring	reflit	Text-only + reflit "value".
slot_tuple_as_ttmd	characterstring	characterstring	multidata	Text-only + multidata "value".
slot_tuple_as_bbb	octetstring	octetstring	octetstring	For binary-only applications.[25]
slot_tuple_as_btb	octetstring	characterstring	octetstring	Binary applications with text "kind".
slot_tuple_as_btmd	octetstring	characterstring	multidata	Binary applications, text "kind", and multidata "value".

EXAMPLE The following are examples of navigable data objects in the UML diagrams below:

self.identifier
self.kind
self.value

13.4.4 slot_tuple

The following is a UML presentation of the **slot_tuple** class:

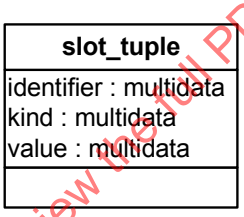

Figure 6 — UML presentation of slot_tuple datatype

13.4.5 slot_tuple_as_ttt

The following is a UML presentation of the **slot_tuple_as_ttt** class:

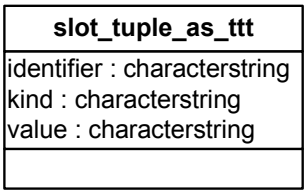

Figure 7 — UML presentation of slot_tuple_as_ttt datatype

24) Text-only applications may store URIs in the characterstring if URIs are agreed upon by data exchange participants.

25) Binary-only applications may store text in the octetstring if the particular text encodings are agreed upon by data exchange participants.

13.4.6 slot_tuple_as_ttrl

The following is a UML presentation of the **slot_tuple_as_ttrl** class:

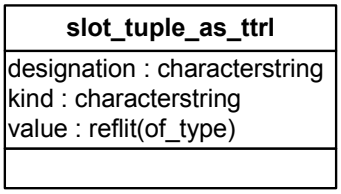

Figure 8 — UML presentation of slot_tuple_as_ttrl datatype

13.4.7 slot_tuple_as_ttmd

The following is a UML presentation of the **slot_tuple_as_ttmd** class:

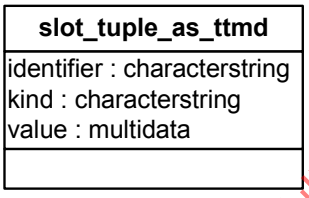

Figure 9 — UML presentation of slot_tuple_as_ttmd datatype

13.4.8 slot_tuple_as_bbb

The following is a UML presentation of the **slot_tuple_as_bbb** class:

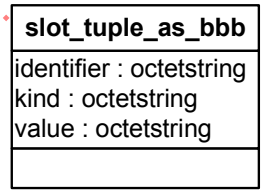

Figure 10 — UML presentation of slot_tuple_as_bbb datatype

13.4.9 slot_tuple_as_btb

The following is a UML presentation of the **slot_tuple_as_btb** class:

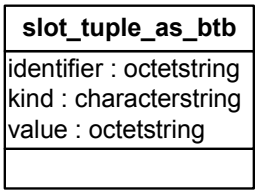

Figure 11 — UML presentation of slot_tuple_as_btb datatype

13.4.10 slot_tuple_as_btmd

The following is a UML presentation of the **slot_tuple_as_btmd** class:

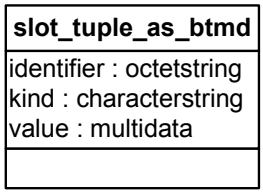

Figure 12 — UML presentation of slot_tuple_as_btmd datatype

13.5 Computational description and datatypes

13.5.1 General

This subclause defines datatypes using ISO/IEC 11404 notation. Provisions embedded in 11404 comments are normative.

13.5.2 Datatypes

Subclause 13.4 defines the datatypes for each of the variants of the slot tuple. The following is a summary of the datatype definitions in ISO/IEC 11404 notation.

ISO/IEC 11404 definition

```
type slot_tuple = record
(
    identifier: multidata, kind: multidata, value: multidata,
),

type slot_tuple_as_ttt = record
(
    identifier: characterstring, kind: characterstring, value: characterstring,
),

type slot_tuple_as_ttrl = record
(
    identifier: characterstring, kind: characterstring, value: reflit(kind),
),

type slot_tuple_as_ttmd = record
(
    identifier: characterstring, kind: characterstring, value: multidata,
),

type slot_tuple_as_bbb = record
(
    identifier: octetstring, kind: octetstring, value: octetstring,
),

type slot_tuple_as_btb = record
(
    identifier: octetstring, kind: characterstring, value: octetstring,
),
```

```
type slot_tuple_as_btmd = record
(
    identifier: octetstring, kind: characterstring, value: multidata,
),
```

13.6 Additional provisions for bindings

There are no additional provisions for bindings.

13.7 Additional provisions for conformity

There are no additional provisions for conformity.

14 Module 14: Data structure for unstructured table of slot tuples

14.1 Introduction to module

Clause 13 (previous Clause) is the data structure that provides the structure of identifier-kind-value (IKV) slot tuples, a mechanism for describing individual data values via the 3-tuple of { **identifier** , **kind** , **value** }.

Clause 14 (this Clause) provides the data structure for bundling an ad hoc table of data values where the identifier is a key for accessing a row of the table.[26)]

14.2 Scope of module

This Clause provides the data structure for bundling a table of unstructured (or structured) data values, based upon the identifier-kind-value (IKV) slot tuple structure defined in Clause 13.

14.3 Functional capabilities

The slot tuple table data structure allows the representation an unstructured table of slot tuples. This Clause does not impose a structure of the data tuples (e.g., choice of names and their nesting), but external specifications can be layered on top of this standard that provide structuring of the data tuples.

EXAMPLE The following examples are from the 11179-3 metamodel:

```
example_11179_data_element : slot_tuple_table_as_ttt =
(
      (
        "data_element_administration.creation_date",
            "time",
                "2004-10-31"
      ),
      (
        "data_element_administration.administrative_note",
            "string",
                "An example note."
      ),
```

26) One common usage is for slot tuple tables is for creating a record-like structure of values whose component names and datatypes are not known in advance. For example, a slot tuple table can be added to a fixed-record structure to provide an mechanism for arbitrary extensions to a record.

```
        (
          "representation_class_qualifier",
              "characterstring",
                  "quantity"
        ),
        (
          "data_element_precision",
              "integer",
                  "2"
        ),
        (
          "exemplified_by.0.data_element_example_item",
              "characterstring",
                  "9.00"
        ),
    ),
```

14.4 Abstract model

14.4.1 General

Conceptually, a slot tuple is 3-tuple that represents a identifier, a kind, and a value (defined in Clause 13). Typically, slot tuples are used in groups (defined in this Clause) and used for general data interchange where datatypes might not be known prior to data interchange. The remainder of this subclause is an object-model description of a slot tuple table and its variants.

14.4.2 slot_tuple_table and related classes

The following is a UML presentation of the **slot_tuple_table** class:

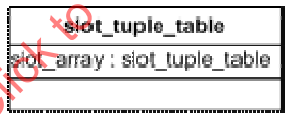

Figure 13 — UML presentation of slot_tuple_table datatype

There are similar UML classes for **slot_tuple_table_as_ttt**, **slot_tuple_table_as_ttrl**, **slot_tuple_table_as_ttmd**, **slot_tuple_as_bbb**, **slot_tuple_as_btb**, **slot_tuple_as_btmd**.

14.5 Computational description and datatypes

14.5.1 General

This subclause defines datatypes using ISO/IEC 11404 notation.

14.5.2 Datatypes

Subclause 14.4 defines the datatypes for each of the variants of the slot tuple. The following is a summary of the datatype definitions in ISO/IEC 11404 notation.

ISO/IEC 11404 definition

```
type slot_tuple_table = array (0..*) of slot_tuple,

type slot_tuple_table_as_ttt = array (0..*) of slot_tuple_as_ttt,
```

```
type slot_tuple_as_ttrl = array (0..*) of slot_tuple_as_ttrl,

type slot_tuple_table_as_ttmd = array (0..*) of slot_tuple_as_ttmd,

type slot_tuple_table_as_bbb = array (0..*) of slot_tuple_as_bbb,

type slot_tuple_table_as_btb = array (0..*) of slot_tuple_as_btb,

type slot_tuple_table_as_btmd = array (0..*) of slot_tuple_as_btmd,
```

14.6 Additional provisions for bindings

There are no additional provisions for bindings.

14.7 Additional provisions for conformity

There are no additional provisions for conformity.

15 Module 15: Data for reified relationships and relationship systems

15.1 Introduction to module

This module specifies the data structures for reifying relationships among objects.

15.2 Scope of module

This Clause provides the data structure for representing data (reifying) about relationships and relationship systems.

15.3 Functional capabilities

The **reified_relationship** and **reified_relationship_system** data structures support reifying relationships, i.e., transforming relationships into data, and support reifying a system of relationships. For example, the data structure can be used to represent relationships among arbitrary sets of objects.

15.4 Abstract model

15.4.1 General

Conceptually, a reified relationship is a relationship kind (e.g., a name or identifier), and a list of objects, each with its own role. The remainder of this subclause is an object-model description.

15.4.2 The reified_relationship_system and the reified_relationship

The following is a UML presentation of the **reified_relationship_system**, **reified_relationship**, and **object_role_pair** classes:

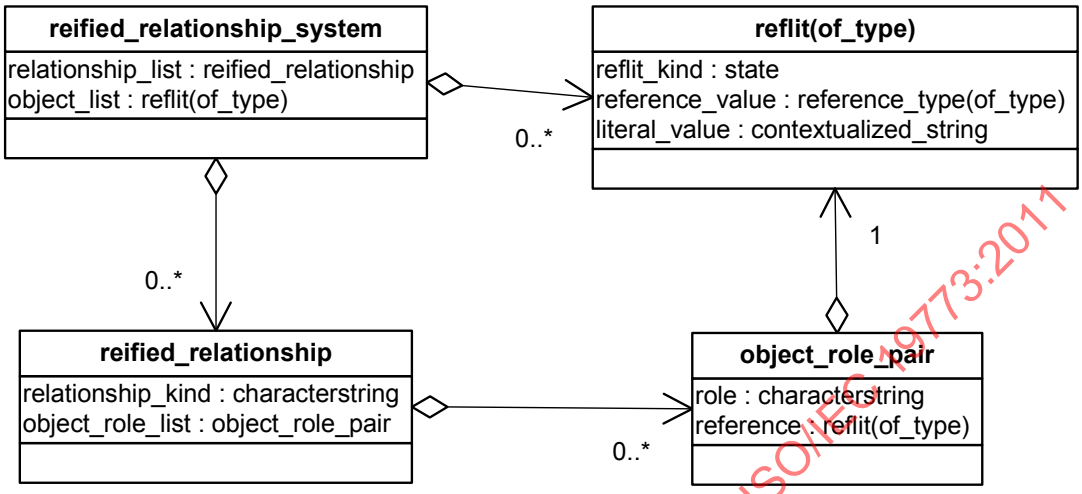

Figure 14 — UML presentation of Reified Relationship Systems

The **reified_relationship_system** class contains the following components:

— **relationship_list : array (0..*) of reified_relationship:** A list of relationships.

— **object_list : array (0..*) of reflit:** A list of objects.

The **reified_relationship** class contains the following components:

— **relationship_kind: characterstring:** A string representing the kind, type, name, or identifier of the relationship kind.

— **object_role_list : array (0..*) of object_role_pair:** A list of objects and their roles.

The **object_role_pair** class contains the following components:

— **role: characterstring:** A string representing the kind, type, name, or identifier of the role the object plays.

— **reference : reflit:** A pointer to the object.

15.5 Computational description and datatypes

15.5.1 General

This subclause defines datatypes using ISO/IEC 11404 notation.

15.5.2 reified_relationship_system

ISO/IEC 11404 definition

```
type reified_relationship_system = record
(
    relationship_list:
```

```
        array (0..*) of reified_relationship,
    object_list:
        array (0..*) of reflit,
),
```

Description

See 15.4.2 for a description of the record and its components.

15.5.3 reified_relationship

ISO/IEC 11404 definition

```
type reified_relationship = record
(
    relationship_kind:
        characterstring,
    object_role_list:
        array (0..*) of object_role_pair,
),
```

Description

See 15.4.2 for a description of the record and its components.

15.5.4 object_role_pair

ISO/IEC 11404 definition

```
type object_role_pair = record
(
    role:
        characterstring,
    reference:
        reflit,
),
```

Description

See 15.4.2 for a description of the record and its components.

15.6 Additional provisions for bindings

There are no additional provisions for bindings.

15.7 Additional provisions for conformity

There are no additional provisions for conformity.

16 Module 16: Data structure for UPU postal data

16.1 Introduction to module

Clause 16 is the data structure that corresponds to Universal Postal Union (UPU) S42a-6 *International postal address components and templates — Part A: Conceptual hierarchy and template languages*. The UPU S42

series specify the structure and semantics of data elements that correspond to postal address elements and postal address segments.

NOTE Although postal addresses are used within the 11179-3 metamodel, they are used within a broader context of Contact Data, which is specified in Clause 19.

16.2 Scope of module

This Clause provides the description of the data structure postal addresses. The UPU S42 address elements are used as a basis for constructing a postal address.

16.3 Functional capabilities

The **postal_address** data structure contains (1) the common international structure of a postal address, and (2) its rendering in one or more country-specific formats. This Clause may be used to present postal address in an unrendered form, or formatted in one or more country-specific formats, or both.

16.4 Abstract model

16.4.1 General

The remainder of this subclause is an object-model description of a **postal address**.

16.4.2 Postal Address

16.4.2.1 postal_address_class

The postal_address_class is comprised of an unrendered postal address and rendered postal address, which may be contextualized in zero or more contexts of use.

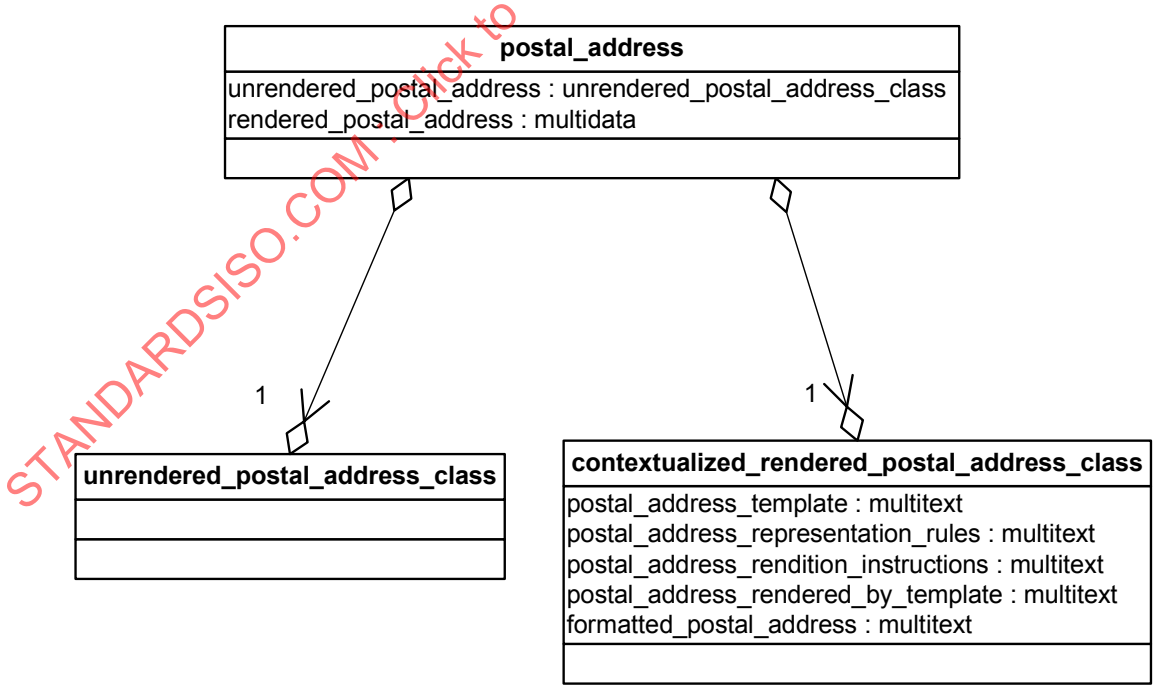

Figure 15 — UML presentation of Postal Address Structure

16.4.3 Unrendered postal data

16.4.3.1 Unrendered postal address structure

The postal address is comprised of segments; segments are comprised of constructs and/or postal address elements; and constructs are comprised of postal address elements.

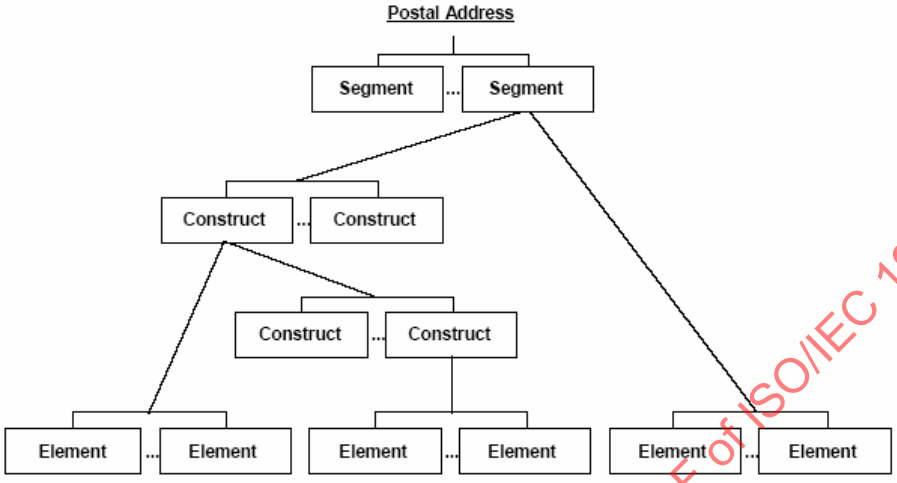

Figure 16 — Postal Address Structure [diagram from UPU S42]

The segments can also be viewed in a lower perspective.

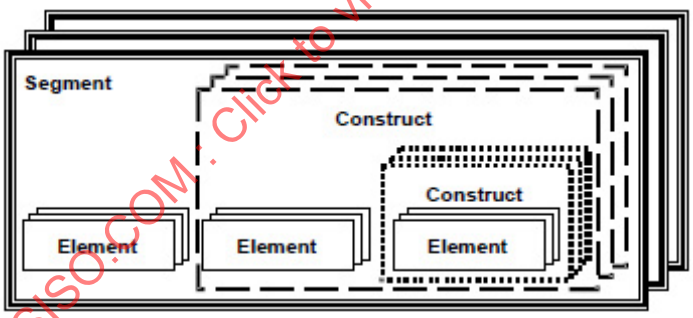

Figure 17 — Perspective of segments, constructs, and postal address elements

The following UML diagram represents the postal address structure above.

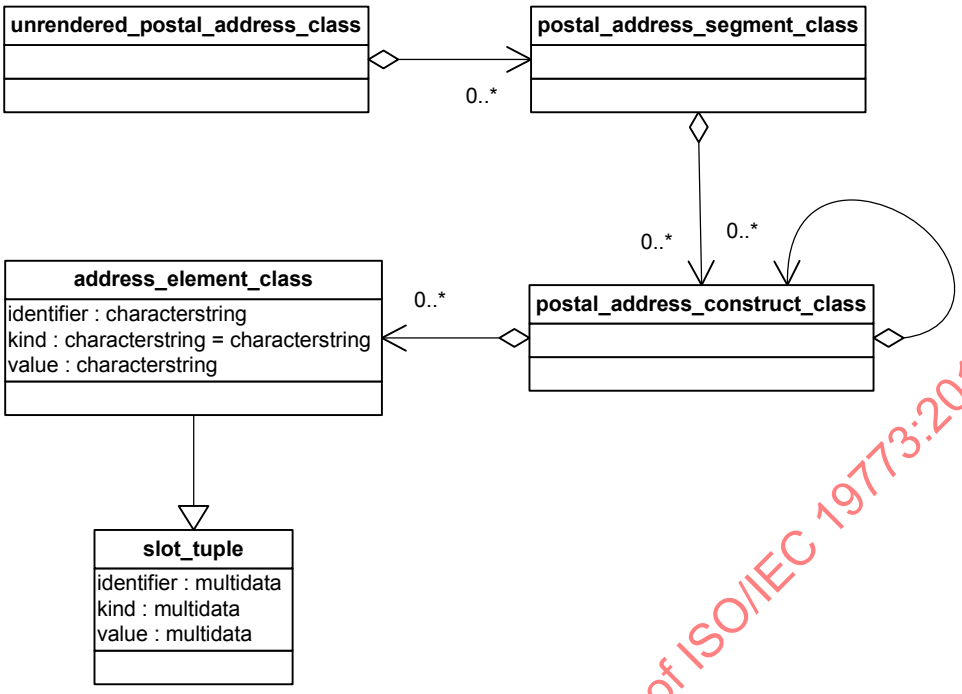

Figure 18 — UML presentation of the classes: unrendered postal address, postal address segment, postal address construct, address element.

16.4.3.2 Postal address components

The unrendered postal address consists of four postal address segments: mailee specification (optional), addressee specification (optional), mail recipient dispatching[27] information (optional), delivery point specification (mandatory).

27) UPU S42a-6 uses both words dispatch and despatch (an alternate spelling according to the Oxford English Dictionary). The OED prefers dispatch, which is used consistently throughout this International Standard.

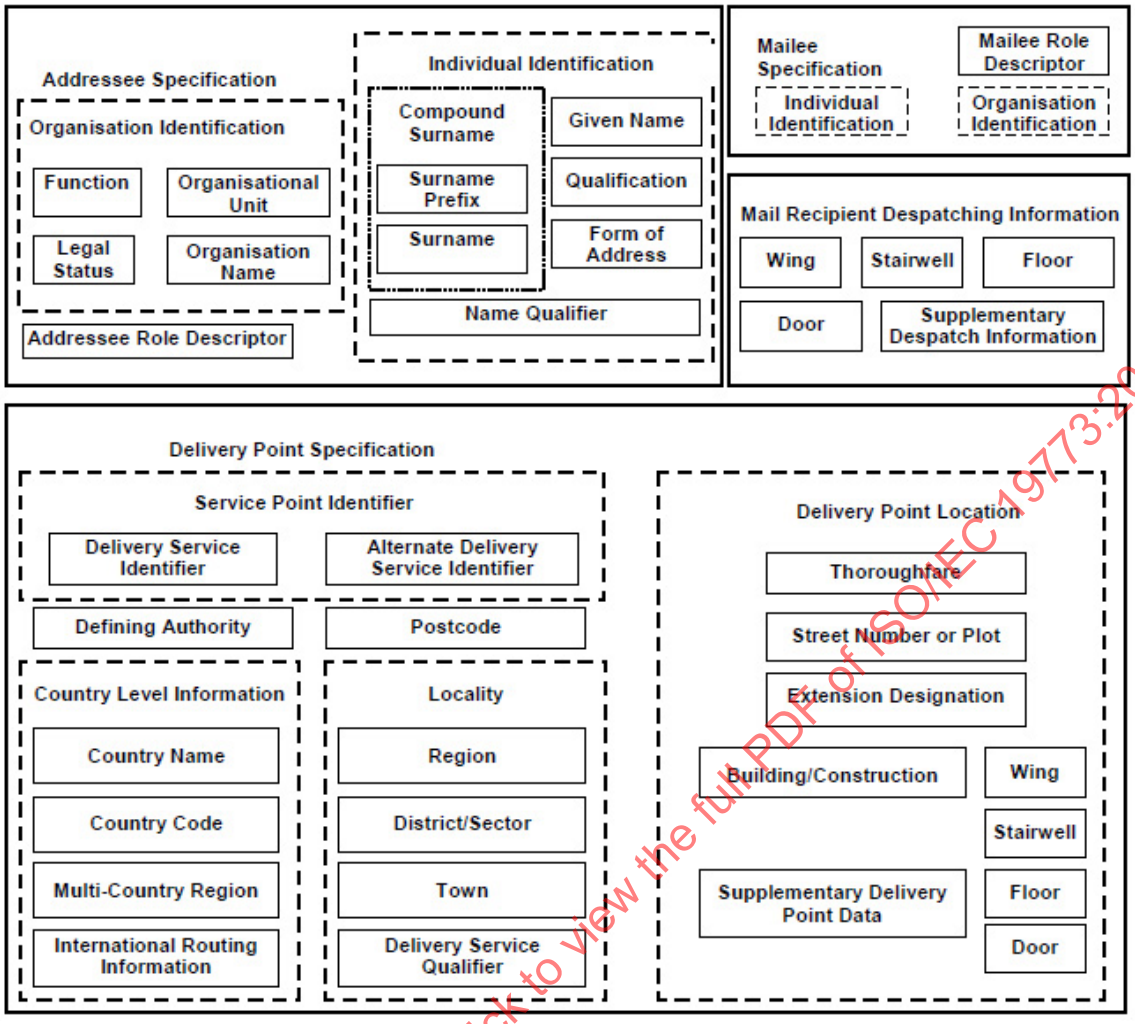

Figure 19 — Postal Address — All Components [diagram from UPU S42]

16.4.3.3 Postal address elements

The unrendered postal address is a nested structure comprised of elements. The following codes correspond to first, second, and third level data identifiers.

In line with its definition, postal address elements are the basic conceptual units from which addresses are built. An element can, however, be present several times and in different locations within the address. As a separate case, addresses within a country may use different locations in the sense of rendition positions for address elements, even if they both are not present at the same time. In addition, address elements, such as region, may have multiple levels in the sense of occurrences within the same address. In UPU S42 multiple occurrences of elements are called instances, in contrast to parts of elements. Different parts of a single element often have to be distinguished during the rendition process, e.g. in order to insert proper punctuation. Instances and parts comprise the two primary sub-divisions of elements, each representing one of the two digits of the element sub-type code. The element sub-types help to represent these multiple instances and parts and facilitate the construction of address templates. When an element is present only once and in an undivided form within an address, so that neither multiple instances nor multiple parts are required, no sub-types are used and the element itself can directly be included within a template. Elements and element sub-types can therefore be considered together as among the building blocks of template.

In forming the element sub-types, there is an issue concerning cardinality. An element such as district/sector may have several levels, positions, or occurrences in an address. Once the need for sub-types is recognized, either they will be limited to some small finite number of instances, or defined with unbounded cardinality. It is not difficult to define an unlimited number of levels of district, for example, by using XML schemas, but in practice the number of levels is limited by the sufficient requirements for unique postal addresses, the constraints of postal databases, and the limited space available for address presentation. Furthermore, defining cardinality as unbounded gives little guidance to designers of address databases. For these reasons, element sub-types are defined in S42 in such a way as to provide enough instances and parts to handle known situations, while providing for some degree of extensibility in implementation.

16.4.3.4 Element codes

An element code comprises two hierarchical levels, the first two digits identifying the segment/construct and the second two digits identifying the element within the segment/construct, the two being separated by a full stop: **xx.yy**

The following identifiers are used in the segment level:

10 addressee specification segment

20 mailee specification segment

30 mail recipient dispatching information segment

40 delivery point specification segment

The second digit of the segment code is reserved for replicating the segment, that is, providing a copy of all the elements and element sub-types within the segment. Where the form of address, for example, is 10.05, a replicated segment provides for a copy with the code **11.05**. This capability provides for multiple addressees with one postal address, or one addressee with multiple postal addresses. In this way it facilitates the design of address databases. However, many postal administrations specify that only one delivery point should be presented in a rendition for mailing purposes. Some postal administrations do allow a post office box delivery point coupled with a thoroughfare address on the same mail piece; an example is South Africa.

The restriction on the use of the second digit of the segment level allows for a maximum of ten segments, of which four are defined. On the element level, the code uses two digits to represent each element, allowing for a maximum of one hundred elements. For S42, thirty-four elements and ninety-eight element sub-types have been defined. This approach keeps the element list relatively compact while providing a capability for high levels of granularity and specificity in storing, combining, and transforming postal address components.

The following identifiers have been assigned to the element level:

00 organization name

01 legal status

02 organizational unit

03 function

04 addressee role descriptor

05 form of address

06 given name

07 surname prefix

08 surname

09 name qualifier

10 qualification

11 mailee role descriptor

12 defining authority

13 postcode

14 country name

15 region

16 town

17 district/sector

19 delivery service identifier

20 alternate delivery service identifier

21 thoroughfare

24 street number or plot

26 building/construction

28 extension designation

29 wing

30 stairwell

31 floor

32 door

33 supplementary dispatch information

34 supplementary delivery point data

35 delivery service qualifier

41 country code

43 multi-country region

44 international routing information

An element sub-type code consists of the code of the element of which it is a sub-type, followed by a hyphen and an identifier consisting of a single digit identifying instances, followed by a hyphen and ending with a single digit identifying parts. The value of each single digit in the element sub-type code is zero if the element sub-type is latent in that dimension, and from one to nine if it is present. As a convention, when using an element directly in a template, the format **xx.yy** and the format **xx.yy-z-z** with each **z** taking the value of zero are considered equivalent.

NOTE The element sub-type "door type" is identified by **40.32-0-1** when it involves information found in the delivery point specification (information used by the postal operator for deliveries). This element sub-type defines a part of door. However, the same element sub-type, but in the mail recipient dispatching information (information intended for the routing and dispatch by the mail recipient, when this is not the addressee), will be identified by **30.32-0-1**.

EXAMPLES **40.13-0-1** is a part of **40.13**. **40.17-1-0** is an instance of **40.17**. **40.17-1-1** and **40.17-1-2** are parts of that instance. **40.17-2-0** is a second instance of **40.17**. **40.13-0-0** is equivalent to **40.13**.

16.4.3.5 Element data identifiers for slot tuples

For postal address elements used within slot tuples, the identifier shall have the following format:

<prefix> _ <postal_element_identifier> _ <element_name>

where **<prefix>** is the string **"int_upu_s42"**, the **<postal_element_identifier>** is described in 16.4.3.4 replacing all punctuation with underscore characters, and the **<element_name>** is described in 16.4.3.4. The following are slot tuple identifiers for the postal elements in the Note in 16.4.3.4:

```
int_upu_s42_40_32_0_1_delivery_point_door_type
int_upu_s42_30_32_0_1_recipient_dispatching_door_type
```

16.4.4 Contextualized Rendered Postal Address

This class holds an rendered postal address. Zero or more renderings of a postal address may be included with an unrendered postal address.

16.4.4.1 contextualized_rendered_postal_address

The **conctentualized_renderend_postal_address** class holds an individual rendering of postal data. This class contains the following components.

— **postal_address_template : multitext:** This component holds the postal addressing template, as defined by UPU S42a-6.

— **postal_address_representation_rules : multitext:** This component defines the data integrity and data transformation rules, as defined by UPU S42a-6.

— **postal_address_rendition_instructions : multitext:** This component defines the rendition instructions, as defined by UPU S42a-6.

— **postal_address_redenred_by_template : multitext:** This component the formatted address, as rendered by the rendition instructions.

EXAMPLE 1 Example of a template for France:

\Line 1\
[form of address] [qualification] [given name] \surname\
OR
\organization name\ [legal status]
[Line 2] [addressee role descriptor] [mailee role descriptor] [form of address] [qualification] [given name] [surname]
OR
[addressee role descriptor] [mailee role descriptor] \organisation name\ [legal status] [organisational unit] [function]
OR
[floor] [door] [supplementary dispatch data]
\Line 3\ [wing] [floor] [door] [supplementary dispatch data]
OR
[building/construction type] [building/construction] [wing] [floor] [door] [supplementary delivery point data]
[Line 4] \street no. or plot\ [extension designation] [thoroughfare type] \thoroughfare name\
\Line 5\ \delivery service type\ \delivery service indicator\ [town]
[Line 6] \postcode\ \town\ [region] [district] [delivery service qualifier]
\Line 7\ \country\.

EXAMPLE 2 Example of representation rules for France:

1. In a geopostal address include the delivery point location information if it is known.

EXAMPLE 3 Example of rendition instructions for France:

1. An address has a maximum of 6 lines (7 for international mail).
2. Only lines that contain information will be printed. If the optional lines are not used there will be no extra space between the lines.
3. An address line has a maximum of 38 characters (including spaces).
4. There must be a space between two words.
5. No punctuation marks, underlining or words in italic are allowed in delivery point location line.
6. Line 6 must always be printed in capital letters and preferably lines 4 to 6.
7. The address blck must be justified to the left.
8. In street numbers 5 bis or 5 ter is shortened to 5 B or 5 T.
9. In the event of a street number with 2 separate sets of figures, only the first number is indicated in line 4 (4 to 8 should written as 4 and 12/14 should be written as 12).

EXAMPLE 4 Example of formatted address:

Formatted address	Address elements
MONSIEUR JEAN DURAND	10.05 MONSIEUR
25 RUE DES FLEURS	10.06 JEAN
33500 LIBOURNE	10.08 DURAND
	14.24 25
	14.22 RUE
	14.21 DES FLEURS
	13.13 33500
	13.16 LIBOURNE

16.5 Computational description and datatypes

16.5.1 General

This subclause defines datatypes using ISO/IEC 11404 notation.

16.5.2 postal_address

ISO/IEC 11404 definition

```
type postal_address = record
(
    unrendered_postal_address:
        unrendered_postal_address_class,
    rendered_postal_address:
        multidata, // multidata contains contextualized_unrendered_postal_class
),
```

Description

See 16.4.2 for a description of the record and its components.

16.5.3 unrendered_postal_address_class

ISO/IEC 11404 definition

```
type unrendered_postal_address_class = record
(
    postal_element:
        array (0..*) of slot_tuple,
),
```

Description

See 16.4.3 for a description of the record and its components.

16.5.4 contextualized_rendered_postal_address_class

ISO/IEC 11404 definition

```
type contextualized_rendered_postal_address_class = record
(
    postal_address_template:
        multitext,
    postal_address_representation_rules:
        multitext,
    postal_address_rendition_instructions:
        multitext,
    postal_address_rendered_by_template:
        multitext,
    formatted_postal_address:
        multitext,
),
```

Description

See 16.4.4 for a description of the record and its components.

16.6 Additional provisions for bindings

There are no additional provisions for bindings.

16.7 Additional provisions for conformity

The following is the 16-21090 profile mapping. In ISO 21090:2011 subclause 7.7.3, the data elements map to the ISO/IEC 19773 module 16 postal data:

— 21090 AL (address line) corresponds to item 34 (supplementary delivery point data)

— 21090 ADL (additional locator) corresponds to secondary items 26 (building/construction), 28 (extension identifier), 29 (wing).

— 21090 UNID (unit identifier) corresponds to secondary items 31 (floor), 32 (door)

— 21090 DAL (delivery address line) corresponds to secondary item 19 (delivery service indicator).

— 21090 DINST (delivery installation type) corresponds to secondary item 20 (alternate delivery service indicator).

— 21090 DINSTA (delivery installation area) corresponds to secondary item 16 (town)

— 21090 DINSTQ (delivery installation qualifier) corresponds to secondary item 35 (delivery service qualifier)

— 21090 DMOD (delivery mode) corresponds to secondary item 19 (delivery service indicator).

— 21090 DMODID (delivery mode identifier) corresponds to secondary item 19 (delivery service indicator)

— 21090 SAL (street address) corresponds to secondary items 21 (thoroughfare name), 22 (thoroughfare type), 23 (thoroughfare qualifier)

— 21090 BNR (building number) corresponds to secondary item 24 (street number or plot).

— 21090 BNN (building number numeric) corresponds to the numeric portion of secondary item 24 (street number or plot)

— 21090 BNS (building number suffix) corresponds to the non-numeric potion of secondary item 24 (street number or plot)

17 Module 17: Data structure for ITU-T E.164 phone number data

17.1 Introduction to module

Clause 17 is the data structure that corresponds to ITU-T E.164 phone number. The data structure may store phone numbers of the international numbering plan (e.g., **+12125551212**) or of private numbering plans (e.g., **0800990011**); the phone numbers may including a PBX (private branch exchange) extension (e.g., **+12125551212+234**, **0800990011+234**).

NOTE The use of non-initial "+" is a structural feature for phone numbers, e.g. in "0800990011+234" the "0800990011" and "234" are separate parts of the full phone number.:

17.2 Scope of module

This Clause provides the description of the data structure phone numbers that may conform to ITU-T E.164. The data structure may store phone numbers of the international numbering plan (e.g., **+12125551212**) or of private numbering plans (e.g., **0800990011**); the phone numbers may including a PBX (private branch exchange) extension (e.g., **+12125551212+234**, **0800990011+234**).

17.3 Functional capabilities

The **phone_number** data structure contains the elements of a phone number. The phone number can be decomposed into several equivalent representations, depending upon the context of use.

EXAMPLE 1 The phone number can +12125551212 equivalent representation (element types are in [] brackets):

E.164 format:
+12125551212 [E.164 character string]
E.164 with punctuation:
+1 212 555 1212 [E.164 character string with punctuation]
E.164 decomposed:
+12125551212 [E.164 character string]
US national format:
(212) 555-1212 [unstructured character string]
New York City local format:
411 [unstructured character string]
France access format:
00 [local international prefix]
1 [country code]
212 [city code]
555 1212 [local number]

EXAMPLE 2 The above structure would be stored in the following phone element structure:

```
nyc_directory_assistance : phone_number_class =
(
      phone_number_element_string  = (
         ( conextualized_value_list =
             ( context_identifier = "ITU-T E.164",
                 value = (
                     ( "ITU-T E.164", "iso-646", "+12125551212" ),
```