caBIG® Platform Independent Model

ISO 21090 Healthcare Data types

Version 0.1.4

Created 26 February 2010
Last Updated 8 June 2010

Contributors*

Todd C. Parnell[1]
John Koisch[2]
Paul Boyes[2]
Prasad Konka[3]
Satish Patel[4]
John Eisenschmidt[5]

Document Version	Author	Changes
0.0.1	Todd C. Parnell	Initial Draft
0.0.2	Todd C. Parnell	Added unmapped types
0.0.3	Todd C. Parnell	Removed classes not currently used in localization.
0.1.0	Todd C. Parnell	Added ED and ED.TEXT mappings.
0.1.1	Todd C. Parnell	PQ.unit and IVL.originalText updated.
0.102	Todd C. Parnell	Added diagrams for each group of types. Added REAL localization.

0.1.3	Abraham Evans-EL	Removing originalText field from QSET and uncertainty, uncertaintyType, and originalText from QTY diagrams
0.1.4	Prasad Konka, Satish Patel, John Eisenschmidt	Removed ED.description and ADXP.INT to be consistent with XSD. Updated ADXP types. Multiple changes to be consistent with model and XSD. Formatting, standardized use of *data type*, corrected captions under figures

[1] 5AM Solutions, Inc
[2] Guidewirearchitecture.com
[3] SAIC
[4] Ekagra Software Technologies
[5] Lantern Three

* The order in which the names are listed do not represent the ownership or importance.

TABLE OF CONTENTS

CABIG® PLATFORM INDEPENDENT MODEL 1

1 INTRODUCTION 4

1.1 RELATIONSHIP TO STANDARDS 4
1.2 RELATION TO CONCEPTUAL FUNCTIONAL MODEL 4
1.3 CONFORMANCE AND COMPLIANCE 4
1.4 NORMATIVE REFERENCES 5

2 DATA TYPES 6

2.1 OVERVIEW AND ARCHITECTURE 6
2.2 TYPE DEFINITIONS 9
2.2.1 Basic Data types 9
2.2.2 Text And Binary Data types 11
2.2.3 Coded Data types (Terminology) 14
2.2.4 Identification and Location Data types 16
2.2.5 Name and Address Data types 19
2.2.6 Quantity Data types 24
2.2.7 Collections Of Data types 28
2.2.8 Continuous Set Data types 29
2.3 UNMAPPED DATA TYPES 30

21090 Platform Independent Model

1 Introduction

The ISO 21090 Healthcare Data types standard "provides a set of data type definitions for representing and exchanging basic concepts that are commonly encountered in healthcare environments in support of information exchange in the healthcare environment, [and] specifies a collection of healthcare related data types suitable for use in a number of health related information environments."[1] This document describes the logical localization of those data types, as used by the associated services.

1.1 Relationship to Standards

ISO 21090 is a draft international standard currently in the Approval stage of the ISO process. The version of the standard this PIM references is ISO/21090 2007-09-24, committee identification ISO/TC 215/WG 2. A newer version of the standard, dated 2009-06-11 "the 2009 version" is available. Where applicable, conformance to the 2009 version is noted when the earlier version had ambiguity. As detailed in Section 1.3, this document details the *Indirect conformance* provided by the individual services.

1.2 Relation to Conceptual Functional Model

No CFM exists for the 21090 data types. However, each of the Conceptual Functional Service Specification (CFSS) documents has an ISO 21010 Data type conformance profile, which references the standard directly.

1.3 Conformance and Compliance

ISO 21090 has two types of conformance: *Direct conformance* and *Indirect conformance*. This PIM supports Indirect conformance, as defined in Section 5.3. Indirect conformance is defined in Section 5.3.1 as:

1. Providing mappings between internal data types and the healthcare data types
2. Specifying for which of the data types an inward mapping is provided, for

[1] ISO/FDIS 21090:2009(E) – Page 1.

which an outward mapping is provided, and for which no mapping is provided

3. Specifying whether the XML representation described herein is used when the data types are represented in XML, or optionally to provide an alternative namespace for the XML representation

Section 2.2 details the inward and outward mappings. Section 2.3 documents data types for which no mapping is provided. Details about the representation are left to platform specific documents.

In addition, Section 5.3.2 requires additional conformance statements:

- Unicode is the character set and encoding for all String[2] types.
- Equality statements are defined at the same time as inward and outward mappings.
- Cardinality applies for each attribute and collection
- Address part type and name bindings, as described in Section 2.2.5.
- All enumeration values retain their meaning in this localization. However, some values have been omitted. The omissions are documented where applicable. If undocumented, all values from each enumeration are used during inward and outward mappings.

1.4 Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this PIM. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based upon this PIM are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

ISO / FDIS 21090:2009(E), *Health Informatics – Harmonized data types for information interchange*
ISO / IEC 8601:2004, *Data elements and interchange formats -- Information interchange -- Representation of dates and times*
ISO / IEC 8824:1990(E), *Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation*
ISO / IEC 11404:2007, *Information technology -- Programming languages, their environments and system software interfaces -- Language-independent data types*

---

[2] ISO 11404 characterstring and UML Kernel String, as applicable.

ISO / IEC 11179, *Information technology — Specification and standardization of data elements*
ISO / IEC 22220, *Health informatics — Identification of subjects of health care*
IETF RFC 1738 - *Uniform Resource Locators (URL)*
IETF RFC 2396 - *Uniform Resource Identifiers (URI): Generic Syntax*
IETF RFC 2806 - *URLs for Telephone Calls*
IETF RFC 2978 - *IANA Charset Registration Procedures*

2 Data types

2.1 Overview and Architecture

For convenience and reference, the following diagram provides an overview in UML Diagrams of the data types defined in this specification.

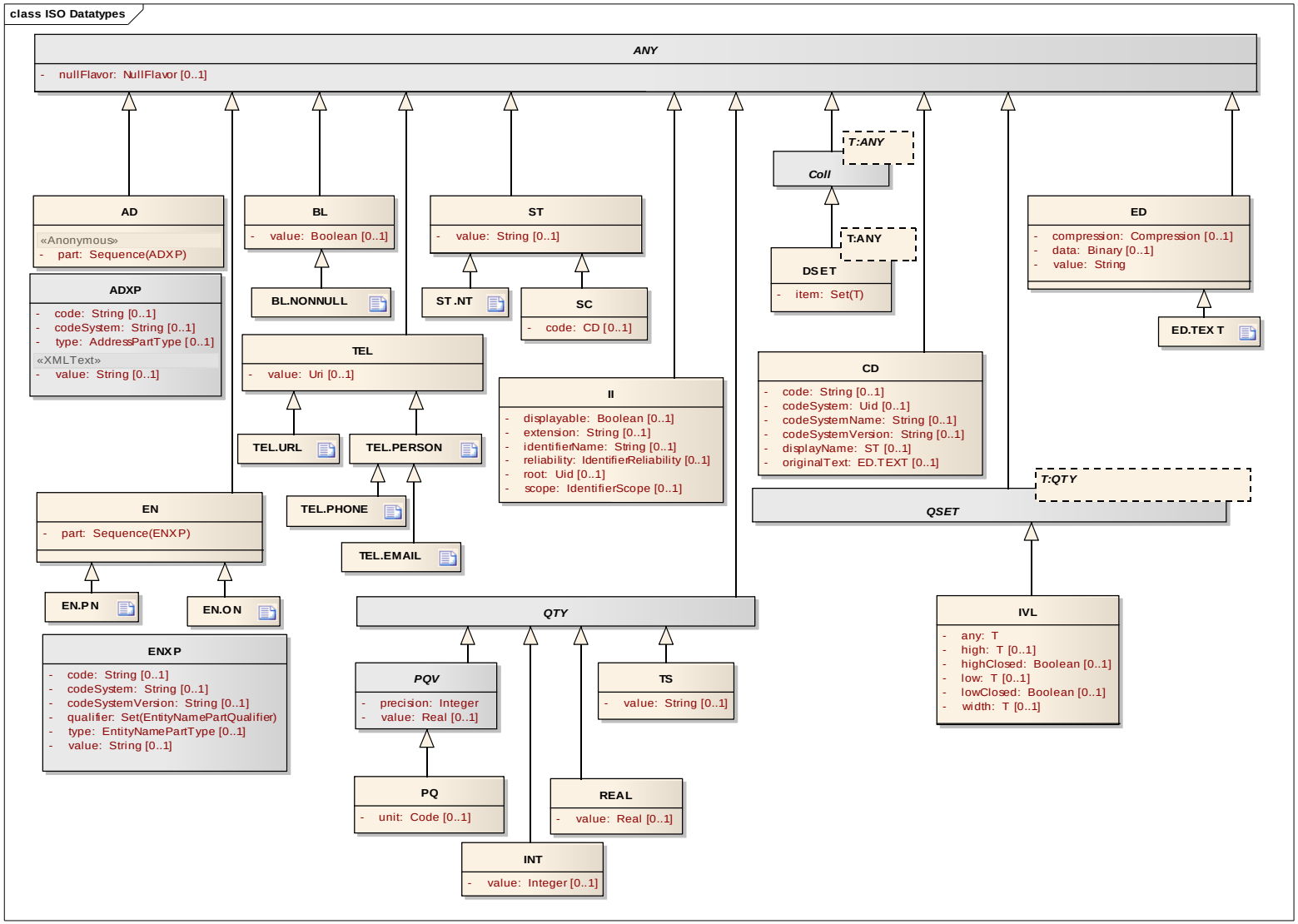

Figure 1 - Localized Classes

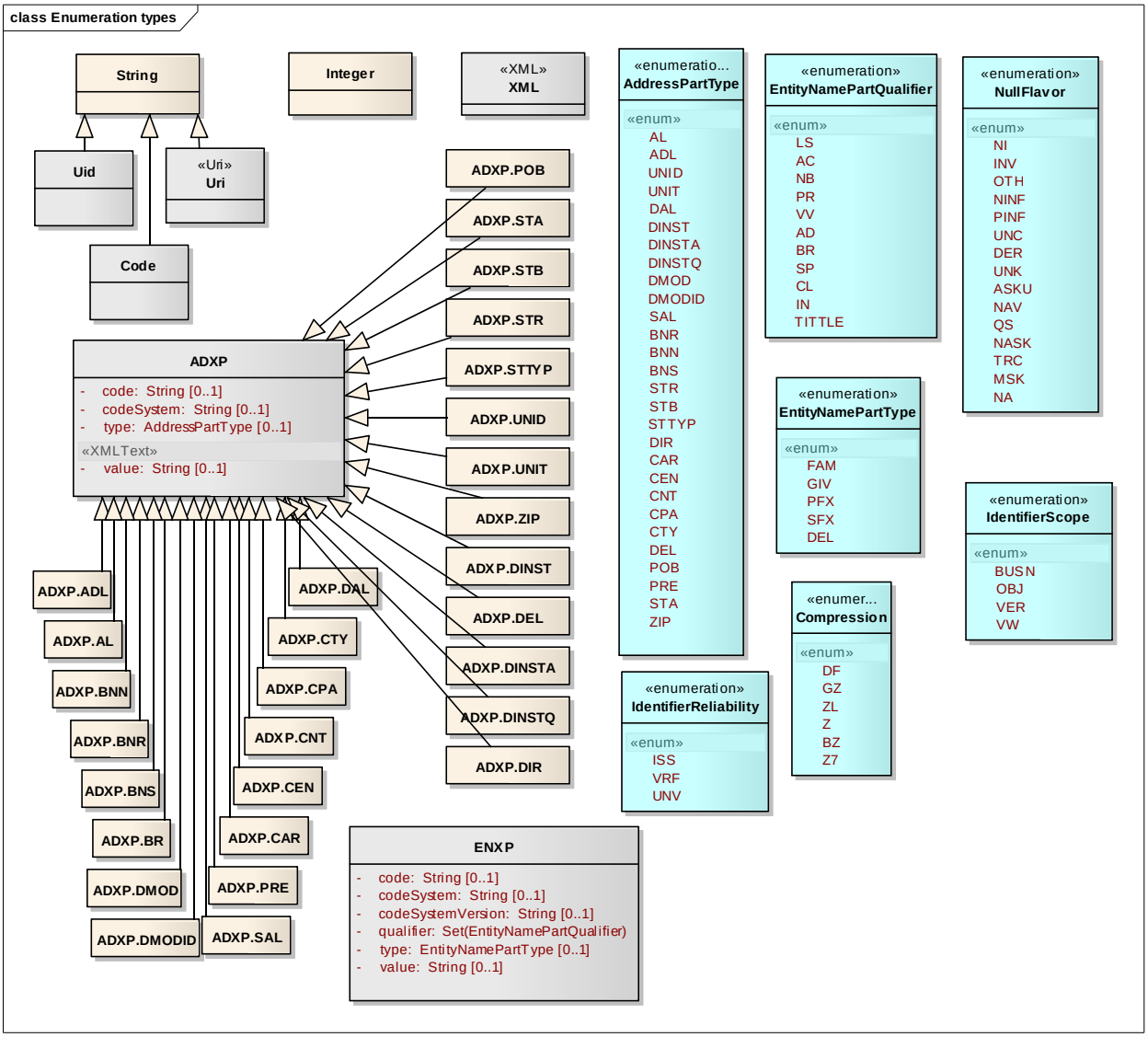

Figure 2 - Localized Enumerations

2.2 Type Definitions

This section details the inward and outward mappings for each data type included in the localization. Data types not included in this section have no mapping and are not part of the localization. Inward and outward transformations of localized data types to ISO 21090 data types is not part of this document. This section focuses on how a localized data type is mapped to ISO 21090 data type inward and outward.

2.2.1 Basic Data types

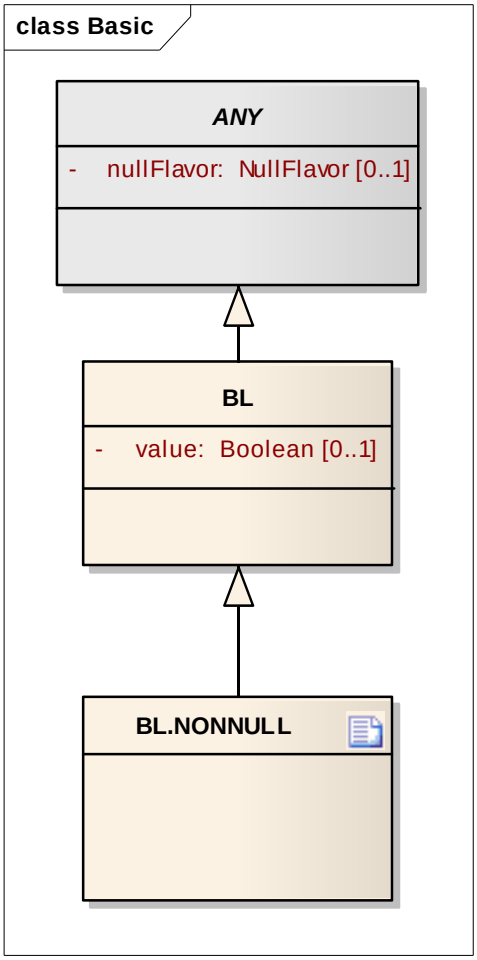

Figure 3 Basic Data types

2.2.1.1 ANY

ANY does not specialize HXIT, as HXIT is not part of the localization.

2.2.1.1.1 ISO 11404 Syntax

```
type ANY = class (
     nullFlavor : NullFlavor
)
```

2.2.1.1.2 Mappings

Inward NullFlavors all map to a single notion of *null* – all enumerated values are treated equally. Outward-*null* values always map to NullFlavor.NI (no information), if a fixed NullFlavor value is not provided.

2.2.1.2 BL

Specializes ANY

2.2.1.2.1 ISO 11404 Syntax

```
type BL = class (
     nullFlavor : NullFlavor,
     value : boolean
)
```

2.2.1.2.2 Mappings

Inward and outward mappings of BL have no special transformation rules.

2.2.1.3 BL.NONNULL

A flavor that constrains BL

There are no additional mapping rules for BL.NONNULL.

2.2.2 Text And Binary Data types

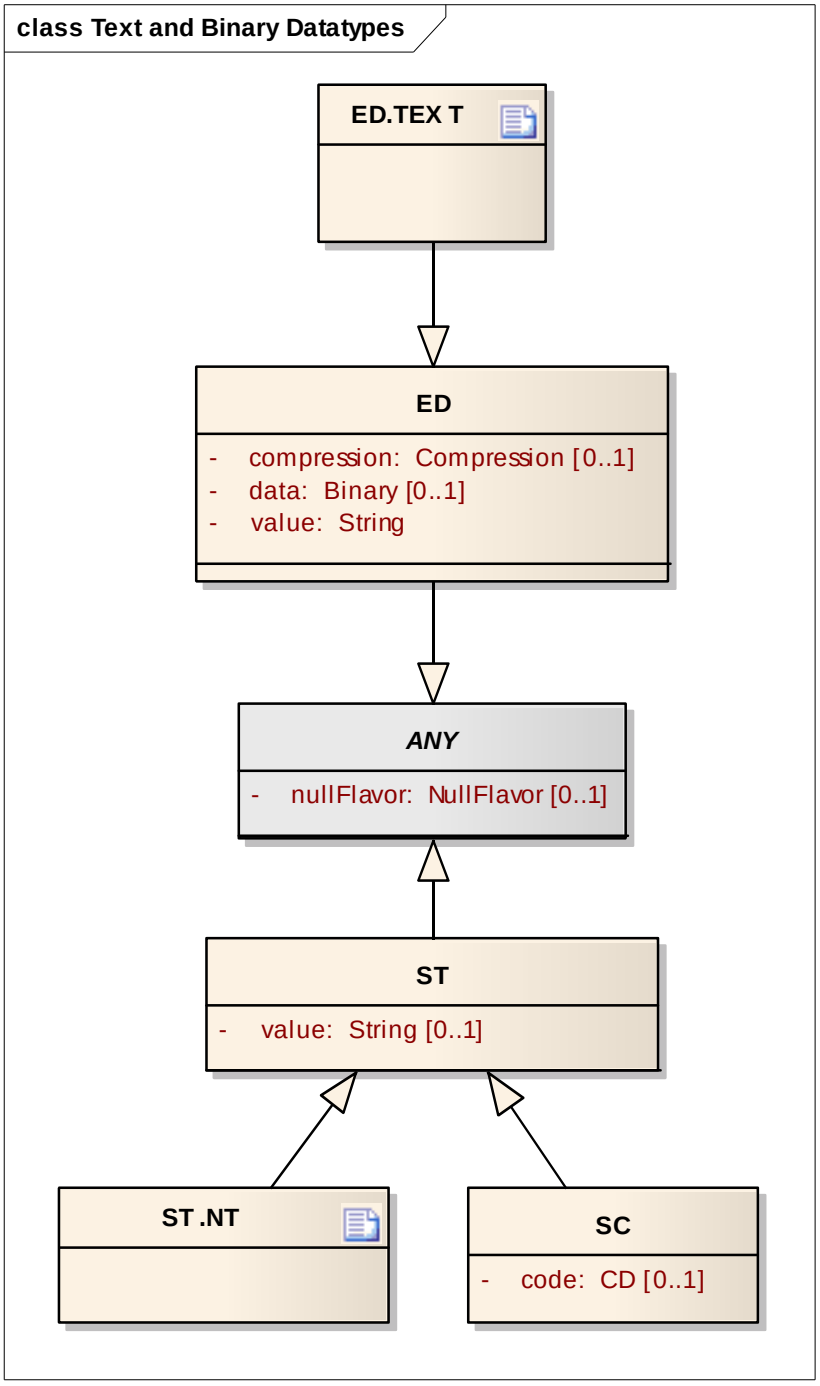

Figure 4 Text and Binary Data types

2.2.2.1 ED (Encapsulated Data)

Specializes ANY.

2.2.2.1.1 ISO 11404 Syntax

```
type ED = class (
     nullFlavor : NullFlavor,
     data: Binary,
     compression : Compression,
     value : characterstring
)
```

2.2.2.1.2 Compression

If compression is NULL the data field is uncompressed. The values understood by this localization are:

CompressionAlgorithm Enumeration localization. OID: 2.16.840.1.113883.5.1009			
Number	Type	Name	Notes
1	GZ	gzip	

Compression types outside this enumeration are not understood and may result in errors during inward mapping.

2.2.2.1.3 Mappings

Inward and outward mappings of ED have no special rules.

2.2.2.2 ED.TEXT

A flavor that constrains ED

There are no additional mapping rules for ED.TEXT. The localization does not contain the xml, reference, integrityCheck, thumbnail, translation elements and mediatype, charset, language, integrityCheckAlgorithm attributes.

2.2.2.3 ST (Character String)

Specializes ANY.

2.2.2.3.1 ISO 11404 Syntax

```
type ST = class (
     nullFlavor : NullFlavor,
     value : characterstring
)
```

2.2.2.3.2 Mappings

Inward mapping has no special rules. The localization does not contain language and translation attributes.

2.2.2.4 ST.NT

A flavor that constrains ST

There are no additional mapping rules for ST.NT. The localization does not contain the translation attribute.

2.2.2.5 SC (Coded String)

Specializes ST

2.2.2.5.1 ISO 11404 Syntax

```
type SC = class (
     nullFlavor : NullFlavor,
     value : characterstring,
     code : CD
)
```

2.2.2.5.2 Mappings

There are no additional mapping rules for SC.

If SC.nullFlavor is null, SC.value is null and SC.code is not null, an error will be generated to set SC.value on non-null SC during inward and outward mapping.

2.2.3 Coded Data types (Terminology)

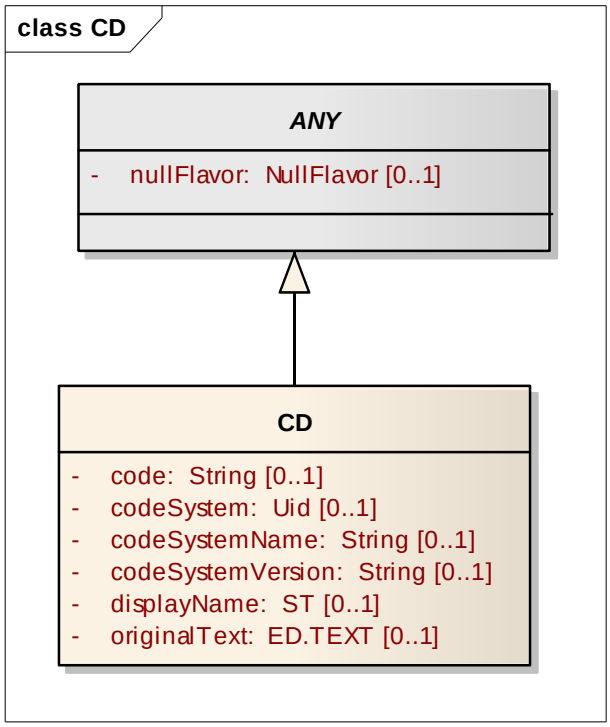

Figure 5 Coded Data types

2.2.3.1 CD (Concept Descriptor)

Specializes ANY

2.2.3.1.1 ISO 11404 Syntax

```
type CD = class (
     nullFlavor : NullFlavor,
     code : characterstring,
     codeSystem : characterstring,
     codeSystemName : characterstring,
     codeSystemVersion : characterstring,
     displayName : ST,
     originalText: ED.TEXT
)
```

2.2.3.1.2 Mappings

There are two types of mapping strategies for CD. Implementations must specify, for each field, which type of mapping is in use.

Type I mappings have fixed internal *codeSystem*, *codeSystemName*, and *codeSystemVersion* values. Thus, during inward mapping, only the *code* itself is stored. The *codeSystem** values should to be identical to the fixed values, though implementations may *ignore* or *reject* variations. During outward mapping, the *code* stored in the system is mapped directly, and the fixed values for *codeSystem, codeSystemVersion, and codeSystemName* are added. Type I implementations must specify the fixed values.

Type II mappings allow variation for *code, codeSystem, codeSystemName,* and *codeSystemVersion.* During inward mapping, each value is stored. During outward mapping, each stored value is mapped directly. Type II implementations may restrict to an enumerated list the valid domain for each attribute. If restricted, the enumerated list must be documented.

The localization does not contain translation, group, source elements and valueSet, valueSetVersion, id, codingRationale attributes.

2.2.4 Identification and Location Data types

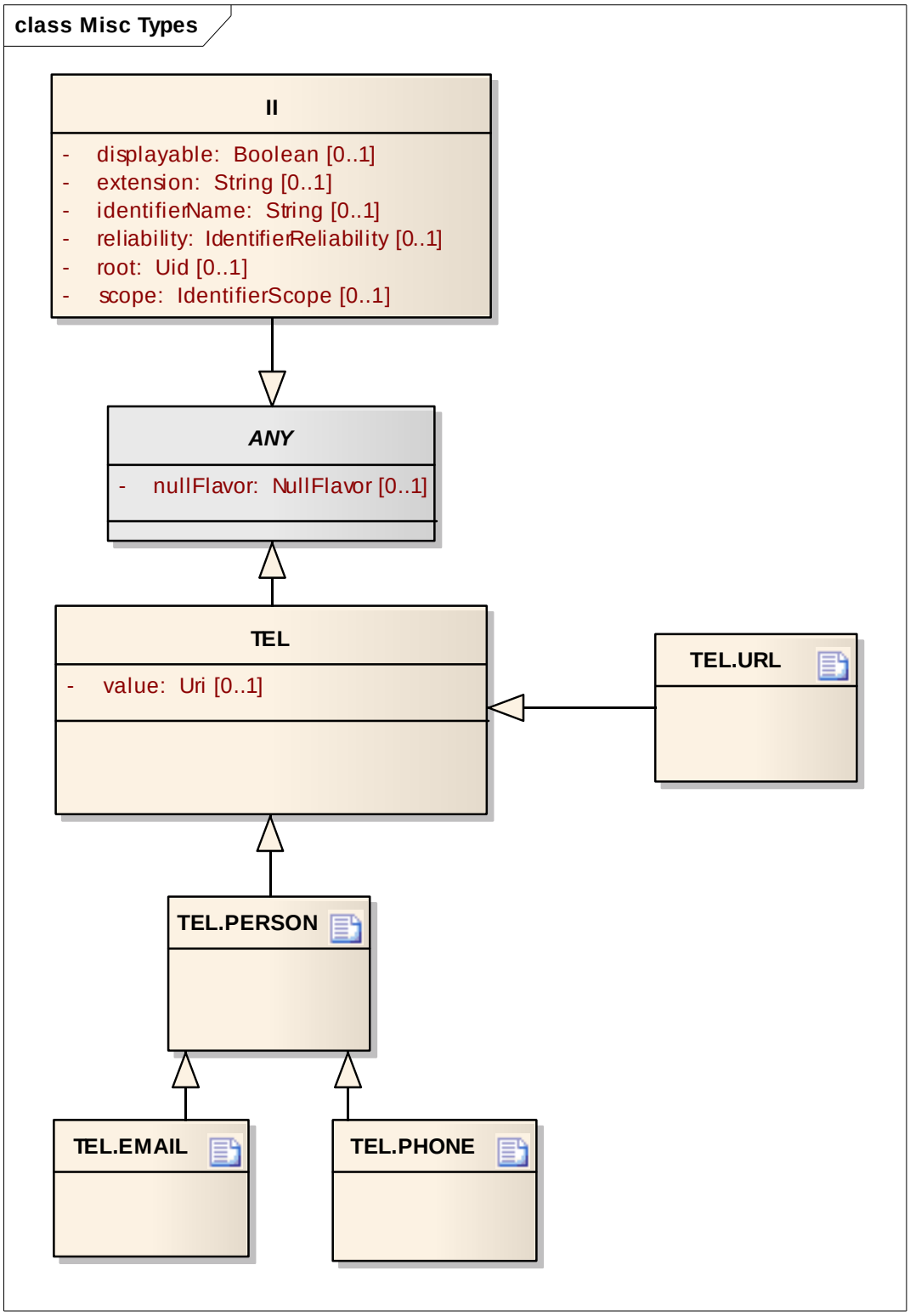

Figure 6 Identification and Location Data types

2.2.4.1 TEL (Telecommunication Address)

Specializes ANY

2.2.4.1.1 ISO 11404 Syntax

```
type TEL = class (
     nullFlavor : NullFlavor,
     value : characterstring
)
```

2.2.4.1.2 Mappings

See the constrained versions of TEL for mapping information.

2.2.4.2 TEL.URL

Constrains TEL

The inward and outward mapping of TEL.URL stores the value attribute directly. An error will be thrown on invalid schema types during inward and outward mapping.

2.2.4.3 TEL.PERSON

Constrains TEL

Inward mapping store only the `<scheme-specific-part>` of the URI, as defined by IETF RFC 2396. The stored value will decode any escaped octets. If the `<scheme>` is unknown, the TEL is treated as NULL with NullFalvor.NI. The outward mapping first escapes (to octets) and characters that cannot be encoded directly in URIs, then adds `<scheme>:` to the escaped `<scheme-specific-part>` value. An error will be thrown on invalid schema types during inward and outward mapping.

2.2.4.4 TEL.PHONE

Constrains TEL

Mappings are specified per TEL.PERSON.

2.2.4.5 TEL.EMAIL

Constrains TEL

Mappings are specified per TEL.PERSON.

2.2.4.6 II (Instance Identifier)

Specializes ANY

2.2.4.6.1 ISO 11404 Syntax

```
type II = class (
     nullFlavor : NullFlavor,
     displayable : boolean,
     extension : characterstring,
     identifierName : characterstring,
     reliability : IdentifierReliability,
     root : characterstring,
     scope : IdentifierScope
)
```

2.2.4.6.2 Mappings

There are two types of mapping strategies for II. Unless otherwise specified, a Type I mapping is assumed.

Type I mappings are "internal id" mappings. For non-NullFalvored instances, *displayable* is always `true`, *reliability* is always `ISS`, and *scope* is always `OBJ if no fixed values are specified by the implementation`. Type I mappings have fixed internal *root* and *identifierName* values. During inward mapping, only the *extension* is stored. *displayable, reliability, scope, root, and identifierName* values should to be identical to the fixed values, though implementations may *ignore* or *reject* variations. During outward mapping, the *extension* stored in the system is mapped directly, and the fixed values for *displayable, reliability, scope, root, and identifierName* are added. Type I implementations must specify the fixed values for *root* and *identifierName.*

Type II mappings are "externally generated identifiers" and allow variation for each attribute. During inward mapping, each value is stored. During outward mapping, each stored value is mapped directly. Type II implementations may restrict to an enumerated list the valid domain for each attribute. If restricted, the enumerated list must be documented.

2.2.5 Name and Address Data types

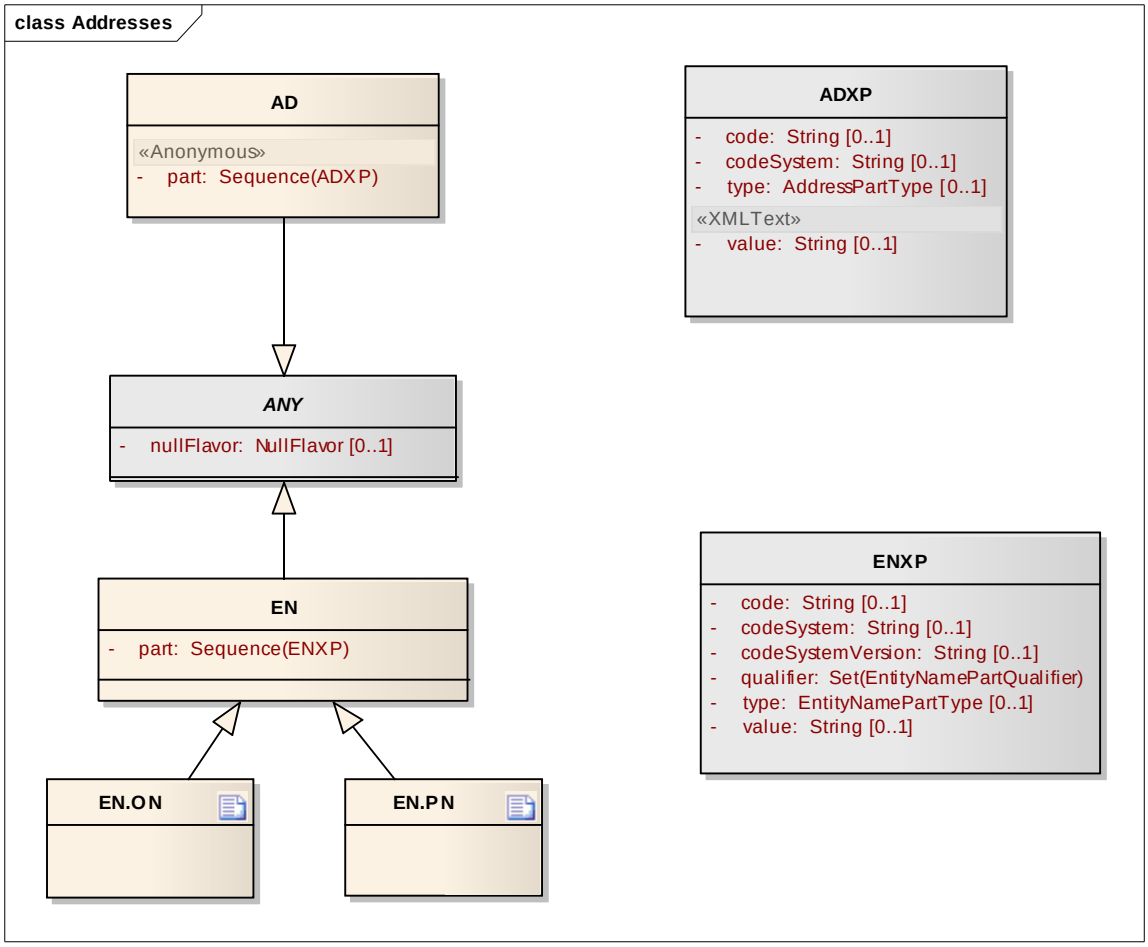

Figure 7 Name and Address Data types

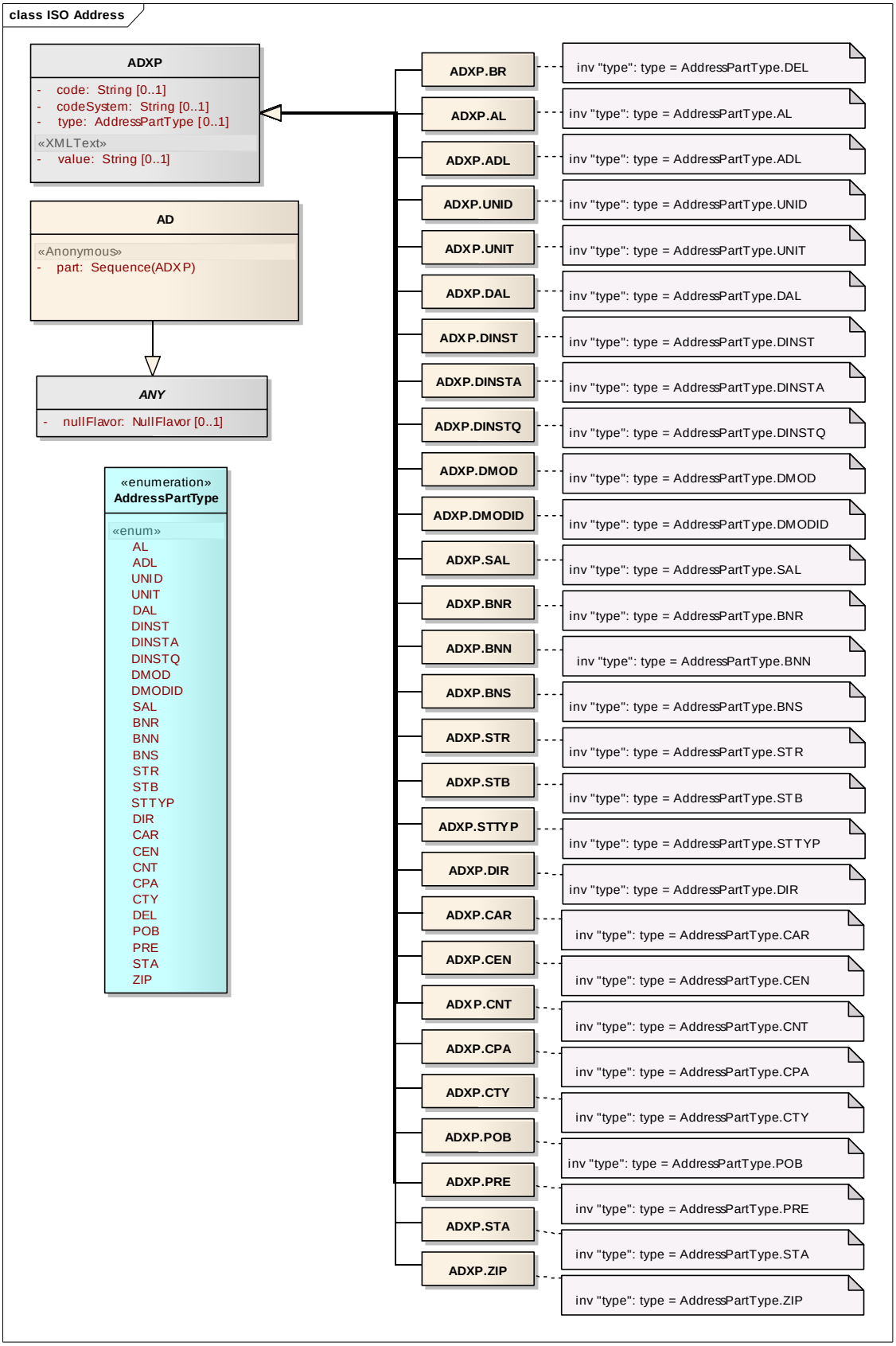

Figure 8 Address Data types

2.2.5.1 ADXP (Address Part)

2.2.5.1.1 ISO 11404 Syntax

```
type ADXP = class (
     value : characterstring,
     code : characterstring,
     codeSystem : characterstring,
     type : AddressPartType
)
```

2.2.5.1.2 AddressPartType

If the type is NULL the address part is unclassified and ignored. The values understood by this localization are:

AddressPartType Enumeration localization. OID: 2.16.840.1.1138883.5.16			
Number	Type	Name	Notes
1	AL	address line	
2	DAL	delivery address line	
3	CNT	country	
4	CTY	municipality	
5	STA	state or province	
6	ZIP	postal code	
7	CAR	care of	
8	ADL	additional locator	
9	DEL	delimiter	
10	POB	post box	
11	BNN	building number numeric	
12	BNR	building number	
13	BNS	building number suffix	
14	CEN	census tract	
15	CPA	county or parish	
16	DINST	delivery installation type	
17	DINSTA	delivery installation area	
18	DINSTQ	delivery installation qualifier	
19	DIR	direction	
20	DMOD	delivery mode	
21	DMODID	delivery mode identifier	
22	PRE	precinct	
23	SAL	street address line	
24	STB	street name base	

25	STR	street name	
26	STTYP	street type	
27	UNID	unit identifier	
28	UNIT	unit designator	

Instances of ADXP with a type outside this localization are ignored.

2.2.5.1.3 Mappings

See Section 2.2.5.2.2. ADXP is always mapped in relationship to the containing AD. This localization does not contain ADXP.codeSystemVersion attribute.

2.2.5.2 AD (Address)

Specializes ANY

2.2.5.2.1 ISO 11404 Syntax

```
type AD = class (
     nullFlavor : NullFlavor,
     part : Sequence(ADXP)
)
```

2.2.5.2.2 Mappings

This localization does not contain useablePeriod, use, isNotOrdered attributes. As an example, systems conforming to this localization can contain the following fields:

- streetAddressLine
- deliveryAddressLine
- cityOrMunicipality
- stateOrProvince
- postalCode
- country

Aside from country, each field is logically a single string value. For country, this is logically a ISO 3116 country.

During outward mapping, NULL internal representations set nullFlavor = NullFlavor.NI (if no NullFlavor fixed value is specified) and part is empty. Non NULL internal representations are converted as follows. Null values for an internal field result in the part sequence not containing an ADXP for that type.

Internal Field	ADXP.type	ADXP.value	ADXP.code
streetAddressLine	AL	*value*	NULL

deliveryAddressLine	DAL	*value*	NULL
cityOrMunicipality	CTY	*value*	NULL
stateOrProvince	STA	*value*	NULL
postalCode	ZIP	*value*	NULL
country	CNT	official country name, per ISO 3166 / MA	three-letter country code, per ISO 3166-1

2.2.5.3 ENXP (Entity Name Part)

2.2.5.3.1 ISO 11404 Syntax

```
type ENXP = class (
     value : characterstring,
     code : characterstring,
     codeSystem : characterstring,
     codeSystemVersion : characterstring,
     type : EntityNamePartType,
     qualifier : Set(EntityNamePartQualifier)
)
```

2.2.5.3.2 Mappings

See Section 2.2.5.4.2. ENXP is always mapped in relationship to the containing EN

2.2.5.4 EN (Entity Name)

Specializes ANY

2.2.5.4.1 ISO 11404 Syntax

```
type EN = class (
     nullFlavor : NullFlavor,
     part : Sequence(ENXP)
)
```

2.2.5.4.2 Mappings

See the mappings section for EN.PN and EN.ON.

2.2.5.5 EN.PN (Person Name)

Constrains EN

As an example, systems conforming to this localization can contain the following fields:

- lastName
- firstName
- middleName
- prefix
- suffix

During outward mapping, NULL internal representations set nullFlavor = NullFlavor.NI (if no fixed NullFlavor value is specified) and part is empty. Non NULL internal representations are converted as follows, in the order specified. Null values for an internal field result in the part sequence not containing an ADXP for that type. By default, any parts missing part type are mapped to GIV type except for a last part in the given list which will be mapped to FAM type.

Order	Internal Field	ENXP.type
1	lastName	FAM
2	firstName	GIV
3	middleName	GIV
4	prefix	PFX
5	suffix	SFX

Inward mapping processes the part sequence in order. Rules:

- type must be non NULL for each part.
- By default, any parts missing part type are mapped to GIV type except for a last part in the given list which will be mapped to FAM type.

2.2.5.6 EN.ON (Organization Name)

Constrains EN

As an example, systems conforming to this localization can have a single field. During outward and inward mapping, a single ENXP is added to part and the *value* is populated from the internal representation.

2.2.6 Quantity Data types

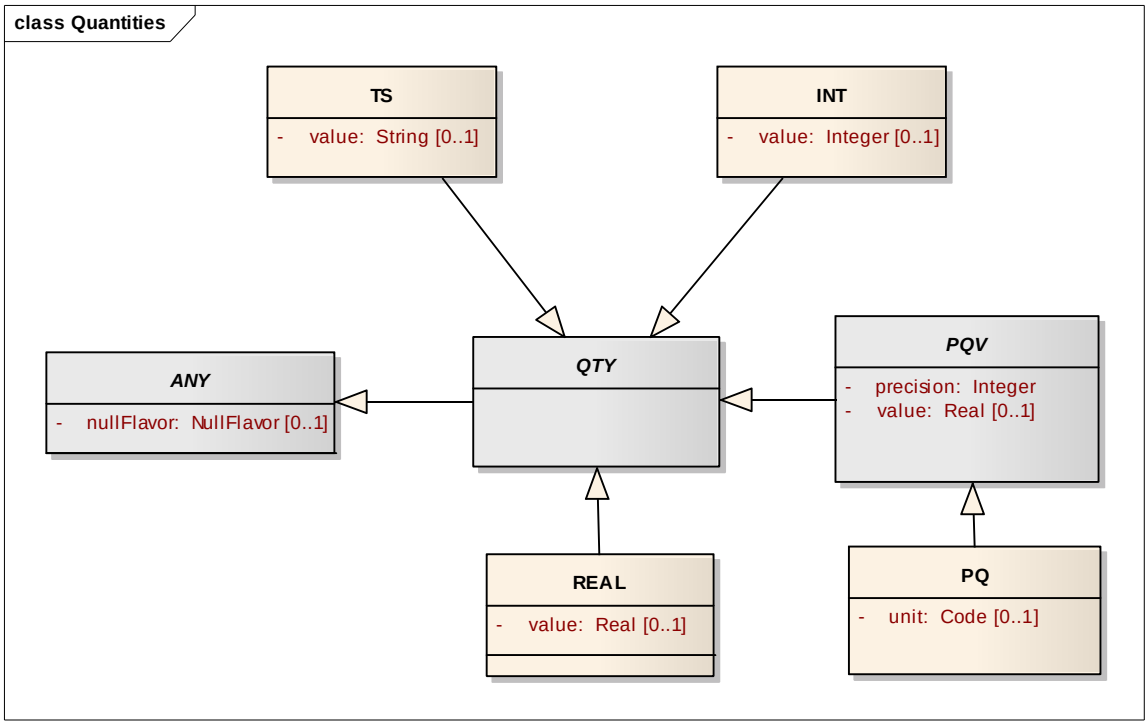

Figure 9 Quantity Data types

2.2.6.1 QTY (Quantity)

Abstract; specializes ANY

2.2.6.1.1 ISO 11404 Syntax

```
type QTY = class (
     nullFlavor : NullFlavor
)
```

2.2.6.1.2 Mappings

See mappings for concrete type implementations.

2.2.6.2 INT (Integer)

Specializes QTY

2.2.6.2.1 ISO 11404 Syntax

```
type INT = class (
     nullFlavor : NullFlavor,
     value : integer
)
```

2.2.6.2.2 Mappings

For inward mapping, NULL instances map to NULL. Non-NULL instances use only the *value* attribute during inward mapping. For outward mapping, NULL maps with *nullFlavor* set to NullFlavor.NI if a fixed NullFlavor value is not specified. For non-NULL instances, *value* will be set to the internal value and all other fields will be empty.

2.2.6.3 PQV (Physical Quantity Value)

Abstract, specializes QTY

2.2.6.3.1 ISO 11404 Syntax

```
type PQV = class (
     nullFlavor : NullFlavor,
     value : real,
     precision : integer
)
```

2.2.6.3.2 Mappings

See section on PQ mappings.

2.2.6.4 PQ (Physical Quantity)

Specializes PQV

2.2.6.4.1 ISO 11404 Syntax

```
type PQ = class (
     nullFlavor : NullFlavor,
     value : real,
     precision : integer,
     unit : Code
)
```

2.2.6.4.2 Mappings

For inward mapping, NULL instances map to NULL. Non-NULL instances use only the *value, precision* and *unit* attributes, storing those attributes directly. For outward mapping, NULL maps with *nullFlavor* set to NullFlavor.NI if a fixed NullFlavor value is not specified. For non-NULL instances, *value, precision* and *unit* will be set to the internal values and all other fields will be empty.

Note: *unit*'s type is Code, which is defined in the 21090 spec as a UNUM code. However, the 21090 XSD treats Code as a xs:string. Internally, this localization

treats unit as a characterstring and does no checking during inward or outward mapping that unit corresponds to a valid UNUM code.

2.2.6.5 TS (Point in Time)

Specializes QTY

2.2.6.5.1 ISO 11404 Syntax

```
type TS = class (
     nullFlavor : NullFlavor,
     value : characterstring

)
```

2.2.6.5.2 Mappings

For inward mapping, NULL instances map to NULL. Non-NULL instances map the *value* attribute per the rules in ISO 8824 (ASN.1) under clause 32 (generalized time). For outward mapping, NULL maps to *nullFlavor* set to NullFlavor.NI if a fixed NullFlavor value is not specified. For non-NULL instances, *value* will be set to the internal value and all other fields will be empty.

2.2.6.6 REAL

Specializes QTY

2.2.6.6.1 ISO 11404 Syntax

```
type TS = class (
     nullFlavor : NullFlavor,
     value : real
)
```

2.2.6.6.2 Mappings

For inward mapping, NULL instances map to NULL. Non-NULL instances map the *value* attribute to an 8 byte IEEE 794 floating-point “double format”. For outward mapping, NULL maps to *nullFlavor* set to NullFlavor.NI if a fixed NullFlavor value is not specified. For non-NULL instances, *value* will be set to the internal value and all other fields will be empty.

2.2.7 Collections Of Data types

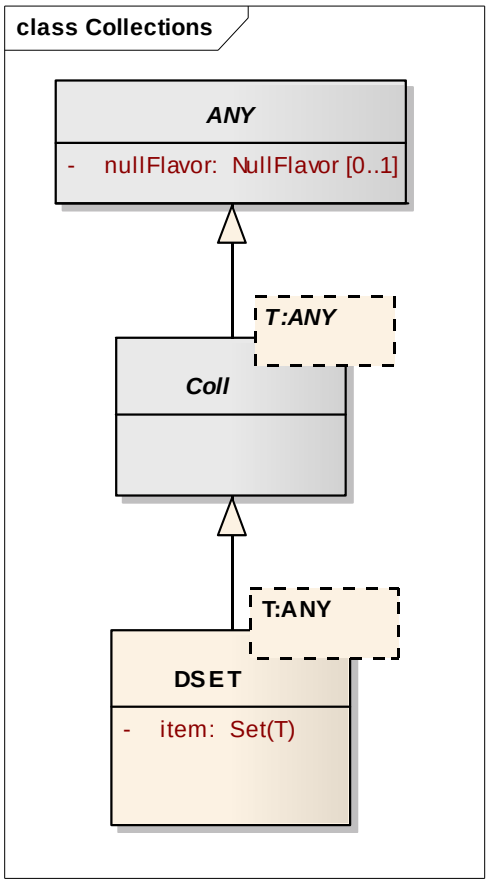

Figure 10 Collections of Data types

2.2.7.1 COLL

Abstract; specializes ANY
Parameter: T : ANY

See next Section for mapping information.

2.2.7.2 DSET (Discrete Set)

Specializes COLL
Parameter: T : ANY

2.2.7.2.1 ISO 11404 Syntax

```
type DSET (T : ANY) = class (
     nullFlavor : NullFlavor,
     item : Set(T)

)
```

2.2.7.2.2 Mappings

During inward mapping, NULL instances map to NULL internal representations. Non-NULL instances map to unordered sets. During outward mapping, NULL or empty sets map with *nullFlavor* set to NullFlavor.NI if a fixed NullFlavor value is not specified. Non-NULL, non-empty sets map directly to the *item* attribute.

2.2.8 Continuous Set Data types

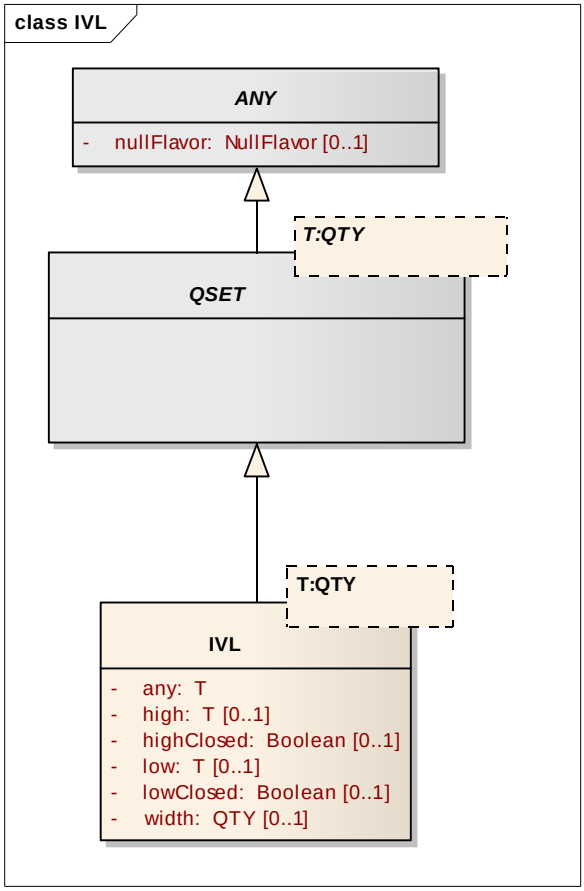

Figure 11 Continuous Set Data types

2.2.8.1 QSET (Continuous Set)

Abstract; specializes ANY

2.2.8.1.1 ISO 11404 Syntax

```
type QSET (T : ANY) = class (
     nullFlavor : NullFlavor
)
```

2.2.8.1.2 Mappings

See next Section for mapping information.

2.2.8.2 IVL (Interval)

Specializes QSET
Parameter: T : QTY

2.2.8.2.1 ISO 11404 Syntax

```
type IVL (T : ANY) = class (
     nullFlavor : NullFlavor,
     any : T,
     high  T,
     highClosed : boolean,
     low : T,
     lowClosed : boolean,
     width : QTY
)
```

2.2.8.2.2 Mappings

This localization introduces any attribute to handle high and low values. During inward mapping, NULL instances map to NULL high, low and width elements. If low value is missing or high value equals low value, any value is set with high value. If high value is missing, any value is set with low value. During outward mapping, if any value is not null and if high and low values are null, any value replaces high and low values. It would also set highClosed and lowClosed to true. During outward mapping, if all the values are null, nullFlavor will be NullFlavor.NI if a fixed value is not specified.

As shown in the localized classes diagram above, localization supports IVL<PQ>, IVL<TS>, IVL<INT>, IVL<REAL> types. IVL by itself is an abstract type.

2.3 Unmapped Data types

The following data types are defined by ISO 21090 but are not part of this localization:

- HXIT
- ED.IMAGE
- ED.DOC
- ED.DOC.INLINE
- ED.DOC.REF

- ED.SIGNATURE
- ED.STRUCTUREDTEXT
- ED.STRUCTUREDTITLE
- CD.CV
- CS
- EN.TN
- INT.NONNEG
- INT.POS
- CO
- RTO
- PR
- PQR
- PQ.TIME
- MO
- TS.DATE
- TS.DATE.FULL
- TS.DATETIME
- TS.DATETIME.FULL
- TS.BIRTH
- LIST (Sequence)
- GLIST (Generated Sequence)
- SLIST (Sampled Sequence)
- HIST (History)
- BAG (Bag)
- QSU (QSET Union)
- QSI (QSET Intersection)
- QSD (QSET Difference)
- QSP (QSET Periodic Hull)
- QSS (QSET Set)
- IVL.CO
- IVL.RTO
- IVL.MO
- PIVL (PerioticInterval)
- EIVL (Event-Related Periodic Interval of Time)
- GTS.BOUNDEDPIVL
- UVP (Uncertain Value – Probabilistic)
- NPPD (Non-Parametric Probability Distribution)
- URG (Uncertain Range)
- URG.LOW
- URG.HIGH
- StructuredDoc.*