Reference number of working document: ?

Date: 2007-09-24

Reference number of document: ?

Committee identification: ISO/TC 215/WG 2

Secretariat: ANSI

Health informatics – Harmonized data types for information interchange

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International standard
Document subtype: Not applicable
Document stage: (40) Enquiry
Document language: E

Copyright notice

This ISO document is a working draft or committee draft and is copyright-protected by ISO and HL7. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

Copyright Manager
American Society for Testing and Materials (ASTM)
100 Barr Harbor Drive, US-West Conshohocken, PA 19428-2959
Tel: +1 610 832 97 31
Fax:+1 610 832 96 66
E-mail: callan@astm.org

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Foreword

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

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

This document is a shared document between Health Level Seven (HL7) and ISO. It is jointly balloted between both organizations. At the ISO level this is currently a committee level document. At the HL7 level, this document is formatted differently to other HL7 V3 specifications because it is a shared document with ISO. So ballot comment should focus on the content rather than the presentation format.

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 21090 was prepared by Technical Committee ISO/TC 215, *Health Informatics*, Subcommittee SC2.3 with assistance from the Infrastructure And Messaging Committee in HL7, and with the support of Connecting For Health.

Contents

HEALTH INFORMATICS – HARMONIZED DATA TYPES FOR INFORMATION INTERCHANGE 1

1 SCOPE 1

2 NORMATIVE REFERENCES 1

3 TERMS AND DEFINITIONS 2

4 ABBREVIATIONS 4

5 CONFORMANCE 4

5.1 Introduction 4

5.2 Direct conformance 4

5.3 Indirect conformance 5

6 DATA TYPES OVERVIEW 6

6.1 What is a datatype? 6

6.2 Definitions of Datatypes 7

6.3 Datatype Names and Reuse of common datatype names 7

6.4 Mapping to this datatypes specification 8

6.5 Conformance to ISO 11404 8

6.6 Reference to UML 2 8

6.7 Modelling of data types 9
6.7.1 Introduction. 9
6.7.2 Generalization / Specialization 9
6.7.3 Enumeration Definitions 9
6.7.4 Strings & Character Encoding 10
6.7.5 Examples 11

7 DATA TYPES 11

7.1 General Properties 12
7.1.1 Immutability 12
7.1.2 Equality 12
7.1.3 History and audit trail 12

7.1.4 Null and NullFlavor 13
7.1.5 Conformance 16

7.2 Top Level Model 17

7.3 Basic Data Types 19
7.3.1 Overview 19
7.3.2 HXIT 19
7.3.3 ANY 21
7.3.4 BL 26
7.3.5 BL.NONNULL 27

7.4 Text And Binary Data Types 28
7.4.1 Overview 29
7.4.2 ED (Encapsulated Data) 30
7.4.3 ED.IMAGE 37
7.4.4 ED.TEXT 39
7.4.5 ED.DOC 39
7.4.6 ED.DOC.INLINE 40
7.4.7 ED.DOC.REF 40
7.4.8 ED.SIGNATURE 40
7.4.9 ED.STRUCTUREDTEXT 41
7.4.10 ED.STRUCTUREDTITLE 42
7.4.11 ST (Character String) 42
7.4.12 ST.NT 44
7.4.13 SC (Coded String) 45
7.4.14 SC.NT 46

7.5 Coded Datatypes (Terminology) 47
7.5.1 CD (Concept Descriptor) 47
7.5.2 CD.CV (Coded Value) 58
7.5.3 CS (Coded Simple Value) 59

7.6 Identification and Location Datatypes 61
7.6.1 TEL (Telecommunication Address) 61
7.6.2 TEL.URL 65
7.6.3 TEL.PERSON 65
7.6.4 TEL.PHONE 66
7.6.5 TEL.EMAIL 66
7.6.6 II (Instance Identifier) 66

7.7 Name and Address Datatypes 73
7.7.1 ADXP (Address Part) 73
7.7.2 AD.Part (Address) 77
7.7.3 AD (Address) 78
7.7.4 ENXP (Entity Name Part) 82
7.7.5 EN (Entity Name) 86
7.7.6 EN.TN (Trivial Name) 89
7.7.7 EN.PN (Person Name) 90
7.7.8 EN.ON (Organization Name) 90

7.8 Quantity Datatypes 91
7.8.1 QTY (Quantity) 92
7.8.2 INT (Integer) 97
7.8.3 INT.NONNEG 99
7.8.4 INT.POS 99
7.8.5 CO (Coded Ordinal) 99
7.8.6 REAL (Real) 101
7.8.7 RTO (Ratio) 105
7.8.8 PQV (Physical Quantity Value) 106

7.8.9 PQ (Physical Quantity) 108
7.8.10 PQR (Physical Quantity Representation) 111
7.8.11 PQ.TIME 112
7.8.12 MO (Monetary Amount) 113
7.8.13 TS (Point in Time) 114
7.8.14 TS.DATE 116
7.8.15 TS.DATE.FULL 116
7.8.16 TS.DATETIME 117
7.8.17 TS.DATETIME.FULL 117
7.8.18 TS.BIRTH 117

7.10 Collection Datatypes 119
7.10.1 COLL 119
7.10.2 DSET (Discrete Set) 120
7.10.3 LIST (Sequence) 123
7.10.4 GLIST (Generated Sequence) 126
7.10.5 SLIST (Sampled Sequence) 128
7.10.6 HIST (History) 132
7.10.7 BAG (Bag) 133

7.11 Continuous Set Data types 135
7.11.1 QSET (Continuous Set) 136
7.11.2 QSU (QSET Union) 139
7.11.3 QSI (QSET Intersection) 139
7.11.4 QSD (QSET Difference) 140
7.11.5 QSP (QSET Periodic Hull) 141
7.11.6 QSS (QSET Set) 142
7.11.7 QSC (Coded QSET) 143
7.11.8 IVL (Interval) 145
7.11.9 IVL.LOW 147
7.11.10 IVL.HIGH 148
7.11.11 IVL.WIDTH 148
7.11.12 PIVL (PeriodicInterval) 148
7.11.13 EIVL (Event-Related Periodic Interval of Time) 151
7.11.14 GTS.BOUNDEDPIVL 153

7.12 Uncertainty Datatypes 155
7.12.1 UVP (Uncertain Value - Probabilistic) 155
7.12.2 NPPD (Non-Parametric Probability Distribution) 156
7.12.3 URG (Uncertain Range) 158
7.12.4 URG.LOW 160
7.12.5 URG.HIGH 160

7.13 Structured Text 161
7.13.2 StrucDoc.Base 164
7.13.3 StrucDoc.Br 165
7.13.4 StrucDoc.Sup 165
7.13.5 StrucDoc.Sub 165
7.13.6 StrucDoc.LinkHtml 166
7.13.7 StrucDoc.RenderMultiMedia 168
7.13.8 StrucDoc.FootnoteRef 169
7.13.9 StrucDoc.Footnote 169
7.13.10 StrucDoc.TitleFootnote 170
7.13.11 StrucDoc.Content 170
7.13.12 StrucDoc.Caption 171
7.13.13 StrucDoc.Captioned 172
7.13.14 StrucDoc.Paragraph 172
7.13.15 StrucDoc.CMFootnotes 173
7.13.16 StrucDoc.CMInline 173
7.13.17 StrucDoc.CMContent 173

7.13.18 StrucDoc.CMGeneral 174
7.13.19 StrucDoc.CMTitle 174
7.13.20 StrucDoc.List 175
7.13.21 StrucDoc.Item 176
7.13.22 StrucDoc.TableItem 176
7.13.23 StrucDoc.TCell 177
7.13.24 StrucDoc.TRow 179
7.13.25 StrucDoc.TRowPart 179
7.13.26 StrucDoc.TRowGroup 180
7.13.27 StrucDoc.ColItem 180
7.13.28 StrucDoc.Col 180
7.13.29 StrucDoc.ColGroup 180
7.13.30 StrucDoc.Table 181
7.13.31 StrucDoc.Text 183
7.13.32 StrucDoc.Title 183

ANNEX A : XML REPRESENTATION 185

A.1 Introduction 185

A.2 Rules for XML Representation 185

A.3 UML / XML schema type mapping 187

A.4 XML Schema 187

ANNEX B : UML SUPPORT TYPES 188

B.1 Introduction 188

B.2 UML data types 188
B.2.1 Octet 188
B.2.2 Binary 188
B.2.3 Code 189
B.2.4 Uid 189
B.2.5 Uri 189
B.2.6 XML 189

ANNEX C : RM-ODP VIEWPOINT MAPPINGS 190

ANNEX D : HL7 V3 ABSTRACT DATATYPES MAPPING 191

ANNEX E : INFORMATIVE SCHEMA FOR XML REPRESENTATION 197

This page is intentionally left blank

Health informatics – Harmonized data types for information interchange

1 Scope

This International Standard

- provides 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,
- specifies a collection of healthcare related data types suitable for use in a number of health related information environments,
- declares the semantics of these data types using the terminology, notations and data types defined in ISO 11404 rev 2005,
- provides UML definitions of the same data types using the terminology, notation and types defined in Unified Modeling Language (UML) version 2.0,
- defines an XML (Extensible Markup Language) based representation of the data types suitable for use when exchanging information between information processing entities.

The requirements which underpin the scope reflected are a mix of requirements gathered primarily from HL7 Version 3 and ISO 11404, and also from CEN 13606 and past ISO work on healthcare datatypes.

Although the focus with regard to requirements and design approaches has focused on the information interchange, these datatypes may be adopted for other purposes in healthcare, but this is not mandated by this standard.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ISO 21090. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this ISO 21090 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. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 4217:2001, *Codes for the representation of currencies and funds*

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 10646-1:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 11404:2005, *Information technology -- Programming languages, their environments and system software interfaces -- Language-independent datatypes*

© All rights reserved

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 1950 - *ZLIB Compressed Data Format Specification version 3.3*

IETF RFC 1951 - *DEFLATE Compressed Data Format Specification version 1.3*

IETF RFC 1952 - *GZIP file format specification version 4.3*

IETF RFC 2045 - *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*

IETF RFC 2046 - *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*

IETF RFC 2396 - *Uniform Resource Identifiers (URI): Generic Syntax*

IETF RFC 2806 - *URLs for Telephone Calls*

IETF RFC 2978 - *IANA Charset Registration Procedures*

IETF RFC 3066 - *Tags for the Identification of Languages*

FIPS PUB 180-1 - *Secure Hash Standard*

FIPS PUB 180-2 - *Secure Hash Standard*

Open Group, CDE 1.1 - *Remote Procedure Call specification, Appendix A*

HL7 V3- *Data Types - Abstract Specification*

Unified Code for Units of Measure - http://aurora.regenstrief.org/ucum

OASIS CIQ - http://www.oasis-open.org/committees/ciq

3 Terms and definitions

For the purposes of this ISO 17113, the following terms and definitions apply

3.1
attribute
characteristic of an object that is assigned a name and a type.

NOTE: The value of an attribute may change during the lifetime of the object.

3.2
class
descriptor for a set of objects with similar structure, behavior, and relationships

3.3
code
symbol that is a reference to a particular concept in a codeSystem

© All rights reserved

3.4
codeSystem
scheme for representing concepts using (usually) short concept identifiers to denote the concepts that are members of the system; defines a set of unique concept codes

EXAMPLES: ICD-9, LOINC and SNOMED

3.5
conformance
fulfillment of a specified requirements; adherence of an information processing entity to the requirements of one or more specific specifications or standards

3.6
data type
set of distinct values, characterized by properties of those values, and by operations on those values

3.7
enumeration
data type whose instances are a set of user-specified named enumeration literals. The literals have a relative order but no algebra is defined on them

3.8
generalisation
generalization is a taxonomic relationship between a more general class, interface or concept and a more specific class, interface or concept. Each instance of the specific element is also an instance of the general element. Thus, the specific element has all the features of the more general element

NOTE 1 The more specific element is fully consistent with the more general element and contains additional information.

NOTE 2 An instance of the more specific element may be used where the more general element is allowed.

3.9
information processing entity
anything which processes information and contains the concept of datatype. This includes other standards, specifications, data handling facilities and services, etc.

3.10
inheritance
mechanism by which more specific elements incorporate structure and behavior of more general elements.

3.11
interface
interface is a specifier for the externally-visible operations of class, without specification of internal structure

3.12
invariant
rule about the features of a class that must always be true

3.13
operation
service that an instance of the class may be requested to perform

Note: An operation has a name and a list of arguments with assigned names and types, and returns a value of the type specified.

© All rights reserved

3.14
specialization
specialization is a taxonomic relationship between a more general class, interface or concept and a more specific class, interface or concept. The more specific entity adds new features or redefines existing features by constraining their possible behaviours

3.15 valueSet
relationship between two classes or concepts whereby one class is completely defined by a subset of another class

4 Abbreviations

The following abbreviations are used for the terms defined in this International Standard and its annexes.

CEN Comité Européen de Normalisation (European Committee for Standardization)

CNE Coded no exceptions

CWE Coded with exceptions

GPD General Purpose Datatypes

HL7 Health Level Seven Inc

IETF Internet Engineering Task Force

OID Object Identifier

OMG Object Management Group

UML Unified Modeling Language

W3C World Wide Web Consortium

XML Extensible Markup Language

5 Conformance

5.1 Introduction

An information processing product, system, element or other entity may conform to this International Standard either directly, by utilizing datatypes specified in this International Standard in a conforming manner, or indirectly, by means of mappings between internal datatypes used by the entity and the datatypes specified in this International Standard.

NOTE: The term information processing entity is used as defined in ISO 11404 Section 4. Specifically, this definition includes applications, and also other standards and specifications.

5.2 Direct conformance

An information processing entity which conforms directly to this International Standard shall:

1. specify which of the datatypes specified in Section 7 are provided by the entity and which are not,

© All rights reserved

2. define the value spaces of the healthcare datatypes used by the entity to be identical to the value spaces specified by this International Standard,

3. to the extent that the entity provides operations other than movement or translation of values, define operations on the healthcare datatypes which can be derived from, or are otherwise consistent with the characterizing operations specified by this International Standard,

4. represent these datatypes using the Extensible Markup Language (XML) representation described herein, when the datatypes are represented in XML

NOTE 1 The requirements above prohibit the use of a type-specifier defined in this International Standard to designate any other datatype (but see the note concerning the scope of the datatype names in 6.2). They make no other limitation on the definition of additional datatypes in a conforming entity.

NOTE 2 Requirement 3 does not require all characterizing operations to be supported and permits additional operations to be provided. The intention is to permit addition of semantic interpretation to the datatypes, as long as it does not conflict with the interpretations given in this International Standard. A conflict arises only when a given characterizing operation could not be implemented or would not be meaningful, given the entity provided operations on the datatype.

NOTE 3 Examples of entities which could conform directly are language definitions or healthcare specifications whose datatypes, and the notation for them, are those defined herein. In addition, the verbatim support by a software tool or application package of the datatype syntax and definition facilities herein should not be precluded.

NOTE 4 Information Processing Entities claiming direct conformance with this standard do not always need to use the datatypes defined in this specification to represent the concepts. I.e. just because an address datatype is defined does not mean that the address datatypes must always be used for representing addresses. However the type defined within this specification SHALL be used where the context is interoperability using these datatypes.

NOTE 5 Information Processing Entities claiming direct conformance with this standard may further constrain the value domain of any of the datatypes within their context of use. The conformance statement must make clear how constraints are applied within the information processing entity, and how values that do not conform to the imposed constraints are handled.

NOTE 6: Consistency of characterizing operations specified by conforming entities may be assessed by these criteria: Where operations have the same name as the operation defined within this specification, they are consistent if the operation can be invoked with the same parameters to return the same result. Other parameters may be defined but must have default values or be defined using additional definitions of operations with the same name but other parameter lists.

NOTE 7: Information Processing Entities claiming direct conformance are not required to call any or all of the types defined in this specification. Other terms such as "Data Structures" may be used.

5.3 Indirect conformance

An information processing entity which conforms indirectly to this International Standard shall:

1. provide mappings between its internal datatypes and the healthcare datatypes conforming to the specifications of Section 7 of this International Standard

2. specify for which of the datatypes in Section 7 an inward mapping is provided, for which an outward mapping is provided, and for which no mapping is provided

© All rights reserved

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

NOTE 1 Examples of entities which could conform indirectly are healthcare specifications, applications, software engineering tools and other interface specifications, and many other entities which have a concept of datatype and an existing notation for it.

NOTE 2 Standards for existing healthcare specifications yet to be proposed as ISO standards are expected to provide for indirect conformance rather than direct conformance.

NOTE 3 Information Processing Entities claiming indirect conformance with this standard do not always need to use the datatypes defined in this specification to represent the concepts. I.e. just because an address datatype is defined does not mean that the address datatypes must always be used for representing addresses. However the type defined within this specification SHALL be used where the context is interoperability using these datatypes.

NOTE 4 Information Processing Entities claiming indirect conformance with this standard may further constrain the value domain of any of the datatypes within their context of use. The conformance statement must make clear how constraints are applied within the information processing entity, and how values that do not conform to the imposed constraints are handled.

NOTE 5: Information Processing Entities claiming indirect conformance are not required to call any or all of the types defined in this specification. Other terms such as "Data Structures" may be used.

6 Data types overview

6.1 What is a datatype?

In ISO 11404, a "datatype" is defined as:

- a set of distinct values, characterized by properties of those values, and by operations on those values
- a datatype consists of three main features: a value space, a set of properties, and a set of characterizing operations

Generally the definitions of the scope of datatypes revolve around either or both of the following notions:

- Immutability (the properties of the data type cannot change, instead a new instance is created: data types have no lifecycle)
- The relationship between equality and identity (if two data types are equal they are the same instance)
- Coherency of a single concept (each data type should represent a single concept space)

Since the application of these concepts to the healthcare information domain and the implications of these for the scope of data types are inherently a matter of perspective, the selection criteria for the data types defined in this standard is based on the set that has emerged from the debates held within the various stakeholder standards bodies that define healthcare information standards. Since healthcare information standards and specifications

© All rights reserved

are expected to provide mappings to this standard, the process has been deliberately inclusive. These other standards may choose to represent these data types with other more complex structures, but should explain how to interconvert these structures with the data types defined herein.

6.2 Definitions of Datatypes

This specification defines a set of named datatypes. Each data type defined in this specification is allocated both a short name and a long name. The formal name of the datatype is the short name. Each data type is defined in two different ways:

- in terms of the data type specification language and types defined in ISO 11404
- in UML using primitive types taken from the UML kernel package

The 11404 definition is provided to ensure continuity between this specification and the 11404 GPDs, while the UML definition is provided to foster software driven implementation of these data types. The 11404 definitions are semantic and abstract in nature, while the UML definitions are concrete structural definitions. This specification is focused on providing structural concrete definitions, so the UML definitions take precedence over the 11404 based definitions, which are provided in the interests of continuity with 11404.

The datatypes defined in this document are an implementation of the HL7 V3 Abstract Datatypes. What this means is that it is possible to implement the exchange of information based on the HL7 V3 abstract datatype definitions using the datatypes defined in this specification. Appendix B demonstrates how these datatypes are implementations of the HL7 V3 Abstract Datatypes.

The datatypes defined in this specification are not restricted to the features described by the HL7 V3 Abstract Datatypes, nor is the HL7 V3 Datatypes Abstract specification required in order to make use of these datatypes. The semantic definitions in the HL7 V3 Abstract Datatypes may be consulted for further useful information to help implementors understand the use of these datatypes.

6.3 Datatype Names and Reuse of common datatype names

Some of the names of these data types bear superficial similarity to similar data types defined in other specifications. For instance, this specification defines a type REAL and there is a type real in the ISO 11404 specification, and a type called Real in the UML kernel.

This specification is not attempting to redefine or replace the definition of real in ISO 11404 or UML. Instead a new type that builds on the functionality of the underlying real types is defined.

There is many specifications, languages and implementation technologies that declare similar types to either the underlying real types or the REAL defined in this specification, with a profusion of names around the common theme of Real, Float or Double.

This specification does not seek to find new previously unrecognised names for the types it declares that represent these base concepts; instead it just assigns the commonly used names to them. Again, this specification is not attempting to redefine or replace any other definitions by using these common names.

For these name clashes between the data types defined in this specification and any other specification, implementers should use some form of namespacing to ensure that the names of the data types do not cause confusion, perhaps by prefixing the names with some string constant in implementation environments that do not support proper namespacing of types.

© All rights reserved

6.4 Mapping to this datatypes specification

Like 11404, this specification anticipates that these datatypes will be used within other specifications. These specifications must specify how the datatypes and features described with this specification are implemented within the specification. Datatypes may be adopted and used directly, or they may be mapped to other datatypes or structures in different places, or they may not be supported at all.

Each specification that uses these datatypes should publish a document or section describing the mapping, and providing assistance for implementors to interconvert data between specifications.

6.5 Conformance to ISO 11404

This specification asserts direct conformance to ISO 11404:2005. Although this specification can be considered to provide support for all the general purpose datatypes, only the following types are actually used in this specification:

- boolean
- enumerated
- characterstring
- integer
- real
- class
- set
- bag
- sequence
- octet

The healthcare datatypes are class constructs built using this base primitive types. The healthcare datatypes are partially defined using the datatype definition language defined in ISO 11404. Since this language does not provide for generalization/specialization relationships, the datatypes cannot be fully defined in the 11404 language, since the generalization/specialization relationships are an important part of the definition of the datatypes.

6.6 Reference to UML 2

This specification defines the datatypes using the UML. The datatypes are all specializations of the UML Classifier, and the types are defined in this specification, or are built on the following UML Kernel types as defined in the OCL 2 specification:

- Enumeration
- Boolean
- Integer
- Real
- String
- Collection
- Sequence
- Set
- Bag

© All rights reserved

6.7 Modelling of data types

6.7.1 Introduction.

This International Standard presents the relationship between individual data types using the facilities of the Unified Modeling Language (UML) version 2. This modelling technique provides a means whereby:

- properties that are common across groups of data types can be expressed once.
- a mechanism is provided whereby one data type within a specification may be substituted by another

6.7.2 Generalization / Specialization

This International Standard defines a series of class representations of healthcare data types. Within these classes, a number of generalization / specialization relationships are defined. This has the normal meaning associated with it as defined in the UML standard, and any instance of a class may be replaced by an instance of a specialization of that class. However some of the specifications that rely upon this standard may make additional constraints concerning which specializations are permissible in a given context.

6.7.3 Enumeration Definitions

This International Standard defines a number of attributes that have an enumerated set of possible values. Each value in the enumeration represents a concept in a terminology. Within the terminology there may be generalization/specialization relationships. In this specification, the enumerations are defined in three ways:

- A list of codes as an 11404 enumeration
- A list of codes as a UML enumeration definitions
- A table defining the enumeration in the narrative of the standard.

The table has four columns, Level, Code, Title, and Definition.

Level	The level of the concept in the generalization/specialization hierarchy of the terminology. All concepts marked with the same level and not separated by a concept with a level of less numerical magnitude are siblings, and all concepts that following after another concept with a higher level value are specializations of that concept.
Code	The code that represents the concept. This is used in the ennumerations and to identify the concept in any representation or exchange of data and in this specification. The Code is indented to represent the generalization/specialization hierarchy of the terminology
Title	A short human readable description of the concept
Description	A short definition of the intension of the concept

The generalization/specialization hierarchy in the enumeration is an important part of the specification. Although the enumerations are defined as linear lists within ISO11404 and the UML definitions, any information processing entity that asserts direct or indirect conformance with this specification must conform to the relationships when evaluating meaning (implication) within the enumeration. In addition, this specification will occasionally refer to

© All rights reserved

the generalization/specialization relationships within the narrative when defining the outcome of some operations.

As an example, here is a subset of the table that defines the NullFlavor Enumeration

NullFlavor Enumeration. OID: 2.16.840.1.113883.5.1008			
Level	Code	Description	Definition
1	NI	No Information	The value is exceptional (missing, omitted, incomplete, improper). No information as to the reason for being an exceptional value is provided. This is the most general exceptional value. It is also the default exceptional value
2	UNK	unknown	A proper value is applicable, but not known
3	ASKU	Asked but unknown	Information was sought but not found (e.g., patient was asked but didn't know)
4	NAV	temporarily unavailable	Information is not available at this time but it is expected that it will be available later
3	NASK	not asked	This information has not been sought (e.g., patient was not asked)
2	MSK	masked	There is information on this item available but it has not been provided by the sender due to security, privacy or other reasons. There may be an alternate mechanism for gaining access to this information. Warning: Using this null flavor does provide information that may be a breach of confidentiality, even though no detail data is provided. Its primary purpose is for those circumstances where it is necessary to inform the receiver that the information does exist without providing any detail
2	NA	not applicable	No proper value is applicable in this context (e.g., last menstrual period for a male)

In this table, all the concepts listed below NI are indented and marked with a higher level, so they are all specialisations of NI. Codes ASKU, NAV and MASK are all specializations of the concept UNK, and codes UNK, MSK and NA are siblings, specialisations of NI but not of anything else. So the eumeration value ASKU implies that the enumeration value UNK is also applicable.

All the enumerations in this specification are maintained by HL7 unless otherwise specified. Revised tables are publised on a regular basis. The values defined in this specification will not have their meaning changed, though they may be deprecated.

The OID for the HL7 codeSystem is published for reference, and to assist when passing these enumerated codes to a terminology sub-systems which are likely to use the OID to unambiguously refer to the code.

6.7.4 Strings & Character Encoding

This specification makes reference to both the ISO 11404 characterstring, and the UML Kernel String. For the purpose of this standard, these types both define the same functionality: an immutable sequence of known lengt containing zero or more logical characters. This type is hereafter referred to in the narrrative of this standard as simply "String".

© All rights reserved

Note: Both ISO 11404 and the UML kernel define additional characterizing operations which may be useful for implementors, and which may be considered to apply, but are not directly of interest to this specification.

Note: This specification also declares a wrapper type for String called ST, which adds additional functionality related to how the notion of an immutable sequence of characters fits into the overall framework of healthcare datatypes with their associated notions to do with uncertainty, unreliability, and conformance.

The String datatype contains a sequence of logical characters, which is different from carrying a sequence of bytes that encodes a sequence of logical characters.

Note: implementers should consider the difference between the two concepts carefully when implementing these datatypes.

By default, the String type contains Unicode characters. Information Processing Entities claiming direct or indirect conformance SHOULD mandate that the unicode character set is used in all String types throughout this specification.

However there a few character sets through out the world that are not perfectly mapped into Unicode. For this reason, there are a few countries or regulatory domains which may mandate some other character set than Unicode. Standards and specifications that claim direct or indirect conformance to this should support of use of character sets other than Unicode.

Note: There is an obvious implementation cost to choosing to use something other than Unicode. Regulatary domains that choose to use something other than Unicode will generally have considered this issue at length.

The character set for any given implementation environment, whether Unicode or something else, is referred to throughout this specification as "the String Character Set“.

Whether the String Character Set is Unicode or something else, any given encoding that serves the implementation of the primitive String type in a particular platform may include control bytes that alter the interpretation of the text. It is assumed that operations performed on the String take this into consideration. However as the operations described in this standard are enacted upon the logical sequence of characters, this issue is not discussed any further.

6.7.5 Examples

Examples are provided for most datatypes. The examples serve to illustrate various points related to how the datatypes are used.

The examples are all given as XML examples following the form documented in Annex A. The examples are presented assuming that the XML document/element that contains them has a charset of UTF-8. Most examples include an xsi:type making their type explicitly clear, but this is generally not necessary where the type is fully specified by the context of use and/or schema.

The discussion in the examples may provide references to content published by standard development organizations other than ISO or HL7. These reference materials are not a normative part of this standard.

7 Data types

© All rights reserved

7.1 General Properties

7.1.1 Immutability

These data types are conceptually immutable. The data types are defined with no lifecycle, and no operations that allow the existing data type to change are defined. However the data types are defined in this specification as classes with attributes, which allows for the value of the datatype to be modified after it is created.

Although this may be useful for implementation, it is not the intent of this specification to define types that are semantically mutable. Specifications that use these datatypes must consider the types themselves immutable. In particular, implementers must be diligent to prevent the aliasing effects that may arise from allowing the datatype properties to change once they are in use.

7.1.2 Equality

There is two aspects to equality – "are these two data values the same instance", or "do these two data values represent the same semantic concept?".

In UML/OCL there are two properties to reflect these two aspects of equality. The first is "=", which returns whether these two values are the same instance, and the second is "equals", which returns whether these types represent the same concept. For OCL primitive types, such as Integer, the two properties return the same value. For UML Classifiers, the results may be different.

In this specification, the equality criteria for each datatype is specified. This specifies the second form of equality: do these two data values represent the same concept. These definitions of equality are carefully constructed to meet the criteria that equals operations must be reflexive, symmetric, and transitive:

reflexive	x equals x must be true
symmetric	If x equals y is true, then y equals x must be true
transitive	If x equals y, and y equals z, then x must equal y

Because this form of equality is inherently a semantic notion, the equality definitions may depend on the semantics of the type, and it may not be simple to determine. Information Processing Entities claiming direct conformance with this standard must conform to these equality definitions. Information Processing Entities claiming indirect conformance must make clear what equality definitions apply.

Datatype flavors do not change the definition of equality.

7.1.3 History and audit trail

The base type HXIT defines properties for specifying a validTime during which the value is, was or will be valid, and for specifying the identity of an event that was associated with the changes (known as a control act).

The validTime is not an audit trail of the information, used to track when any particular systems associated particular versions of data with a concept; it is used to make statements about the time period during which the data item was a correct description of the concept. For instance, in many countries a person may change their name by marriage or other legal means, and so a particular name is only associated with a person for a limited time. Similarly, a persons addresses and contact details may change as they move.

The control act is the identity of some event in an information system which is associated with “control” information concerning the change to this data value on a system. This information

© All rights reserved

specifically relates to the association of data with it's concept in systems. The control act reference can be used to build an audit trail of values across multiple exchanges between systems.

The various specifications that make use of this specification may provide alternative mechanisms for specifying this history and audit data, particularly the second part relating to system audit trails. In such cases, the specification will declare, usually in it's conformance statement, how such information is handled and potentially mapped into the properties defined in this specification.

7.1.4 Null and NullFlavor

The base type ANY introduces a concept called nullFlavor. Though the nullFlavor concept has some relationship with the UML/OCL null, it is not the same thing, and the relationship and differences between the two must be understood to properly implement this specification.

Any instance of a class defined in this specification may be null. This is the null defined in UML and OCL. A null instance is an instance of the type OclVoid, and conforms to all types. It carries no other information other than the fact it is null. This specification uses the ocl operations oclIsDefined and oclIsUndefined to make constraints on the use of this form of null in attributes of the types defined in this specification.

Alternatively, an instance of the class may be created, and it's nullFlavor can be set to one of the NullFlavors.

NullFlavor Enumeration. OID: 2.16.840.1.113883.5.1008			
1	NI	no information	The value is exceptional (missing, omitted, incomplete, improper). No information as to the reason for being an exceptional value is provided. This is the most general exceptional value. It is also the default exceptional value
2	INV	Invalid	The value as represented in the instance is not a member of the set of permitted data values in the constrained value domain of a variable
3	OTH	Other	The actual value is not a member of the set of permitted data values in the constrained value domain of a variable. (e.g., concept not provided by required code system)
4	PINF	positive infinity	Positive infinity of numbers
4	NINF	negative infinity	Negative infinity of numbers
3	UNC	unencoded	No attempt has been made to encode the information correctly but the raw source information is represented (usually in originalText)
3	DER	derived	An actual value may exist, but it must be derived from the provided information (usually an expression is provided directly)
2	UNK	unknown	A proper value is applicable, but not known
3	ASKU	asked but unknown	Information was sought but not found (e.g., patient was asked but didn't know)
4	NAV	temporarily unavailable	Information is not available at this time but it is expected that it will be available later
3	NASK	not asked	This information has not been sought (e.g., patient was not asked)
3	QS	sufficient quantity	The specific quantity is not known, but is known to be non-zero and is not specified because it makes up the bulk of the

© All rights reserved

			material.'Add 10mg of ingredient X, 50mg of ingredient Y, and sufficient quantity of water to 100mL.' The null flavor would be used to express the quantity of water
3	TRC	trace	The content is greater than zero, but too small to be quantified
2	MSK	masked	There is information on this item available but it has not been provided by the sender due to security, privacy or other reasons. There may be an alternate mechanism for gaining access to this information. Warning: Using this null flavor does provide information that may be a breach of confidentiality, even though no detail data is provided. Its primary purpose is for those circumstances where it is necessary to inform the receiver that the information does exist without providing any detail
2	NA	not applicable	No proper value is applicable in this context (e.g., last menstrual period for a male)

In this case the instance is referred to as nullFlavored, or exceptional. The intent of this is that the instance represents a value that is an exception to the normal value domain of the type. This does not mean that an instance representing exceptional value is not bound by the rules defined in this specification; it must still meet the invariants defined herein. However many of the rules are different for exceptional values as they represent semantic exceptions to the normal data. All exceptional values must have a nullFlavor, and the nullFlavor provides more information as to why the value is an exception to the rules. The opposite of an exceptional value is a proper value: this is a value with no nullFlavor.

Since an instance of the type must be created in order to carry a nullFlavor, it can also have values assigned to it's other attributes. The invariants defined in this specification define which attributes may or are required to be populated when a nullFlavor is assigned. This is different from the UML/OCL null which does not exist and cannot have any nonNull attributes.

ISO 11404 defines the concept of a sentinel value, which is a value in the value space of the type that does not share in all the characterizing operations of the type. Though there is some conceptual similarities between nullFlavors and sentinel values, instances of type ANY with nullFlavors are not sentinel values; the characterising operations still apply, though the result of these characterising operations will be some flavor of null. This is like the OCL null behavior, behavioral similarity is why the property is named "nullFlavor".

Unless specifically documented otherwise, all operations defined in this specification behave in the same fashion:

- If the operation is performed on a UML/OCL null value, the result is a UML/OCL null.
- If the operation is performed on a value with a nullFlavor:
    - If the operation takes no parameters, it will return a nullFlavor. Usually the nullFlavor will be NA, but other nullFlavors may be appropriate, depending on the semantics of the nullFlavors. When performing operations upon null values, the semantic meaning of the nullFlavor SHALL be considered.
    - If the operation takes parameters, and any of the parameters are a UML/OCL null, the result will be a UML/OCL null

© All rights reserved

    - If the operation takes parameters, and any of the parameters are a nullFlavor, the result of the operation will be the first common generalisation of the two flavours

- If the operation is performed on a proper value
    - If the operation takes parameters, and any of the parameters are a UML/OCL null, the result will be a UML/OCL null
    - If the operation takes parameters, and any of the parameters are a nullFlavor, the result of the operation will be the first common generalisation of the nullFlavors encountered
    - Otherwise, the operation will perform as described.

Some specific operations deviate from this rule. This is documented for each operation.

One special case arises with equality comparisons of various null values. Two values that have nullFlavor NotApplicable are considered equal. While NINF cannot be equal to NINF and PINF cannot be equal to PINF, since the actual value is not known, NIF and PINF are clearly not equal. In other cases, it is generally not safe for the comparison to return anything but a nullFlavor – usually NI.

A value of any type with a nullFlavor of NI and where all the other attributes are null or meet this rule recursively is semantically equivalent with a UML/OCL null, and these forms may be interconverted if desired. Most simple attributes are declared using a UML or OCL type such as String, and these may be converted to the complex equivalent, such as ST, with a nullFlavor of NI, if desired.

Because an attribute may be either null or a nullFlavor, many of the invariants take the form (x.oclIsDefined and x.isNotNull) implies {condition}. For some invariants it is not necessary to make this rule since the result of null (the value of x.isNotNull if x is actually null) implies anything is null, and the invariant will fail gracefully, but for others the null must be protected against in order for the correct outcome in all cases.

The nullFlavor concept provides a general framework for handling incomplete data which is often encountered in healthcare information collection, use and analysis. The *nullFlavor* property may also play a special role in conformance frameworks in specifications that make use of these types.

Not all the nullFlavors can be used with all the different types. The nullFlavors PINF and NINF may only be used with specific types (INT, REAL, PQ and TS). The nullFlavor UNC may only be used with types that have an originalText (CD, QTY, QSET, and specializations). The nullFlavors QS and TRC may only be used with PQ.

Note: the two nullFlavors INV and OTH draw a distinction between the actual value and the vlaue as represented in the instance. Some of the datatypes may be used to provide a representation of the value which requires subsequent transformation to generate the real value. For instance, an expression may be provided which will generate an actual value that is in the required value domain of the instance.

Note: although INV and OTH represent exceptional values, they are only exceptional within the parameters defined by this specification. The instance SHALL always conform to the rules defined by this specification.

Note: Most often, the correct nullFlavor to use is NI, and there need not be any burden on implementors with regard to choosing or persisting the correct nullFlavor. Other than some technical requirements that are clearly document in this standard regarding the use of the nullFlavors NA, INV, DER, OTH, PINF and NINF, whenever it is not clear which nullFlavor is applicable, implementers should be comfortable using the nullFlavor NI. In particular, if a user

© All rights reserved

does not respond to an input field in a data collection procedure, or the data is missing for some unknown reason, NI would be the most appropriate nullFlavor to choose.

Examples

Use Case	NullFlavor of Choice
User does not respond to input on a screen form	NI
Source is unable to encode plain text input to required codeSystem	UNC
Source is unable to encode this particular plain text input to the required codeSystem because it cannot match the text	OTH
Patient is unconscious and cannot provide name	NAV
The system does not support this element	NI
No proper dosage is provided, but an expression is provided so the destination system can calculate the proper dosage from the patient's weight	INV
The source system is responding to a query for patient details, and has decided not to include the address because of applicable security and/or privacy policy. Note that normal security/privacy policy is not to inform the information recipient that information has been suppressed, for good reason. However there are a few cases where workflow reasons dictate that it is necessary to inform the user that information has been denied. The MSK nullFlavor is provided for these minor cases.	MSK

7.1.5 Conformance

Conformance as discussed in section 5 (Conformance) is concerned whether information processing entities conform to this specification. There is an additional aspect, which is also often called conformance, which concerns what rules are applied to the datatypes by information processing entities such as other standards when the datatypes are used.

Any information processing entity that uses these datatypes may constrain their use by making some human language narrative statements concerning how they are used, or by using some formal language statement in a processible language such as OCL. In addition, this specification recognises two additional means by which the possible values of the datatypes can be constrained, called "mandatory" and "cardinality".

Any external attribute which is assigned a type from this specification may also have a nominal flag "mandatory" set to true. If the context of use sets this flag to true, the instance SHALL contain a valid data value that is not null, has no nullFlavor, conforms to all the constraints stated in this specification and any additional constraints on the value domain stated in the model. If this flag is not set to true, and the instance does not meet the constraints specified in the Constraining Model, the instance SHALL be labelled as null by using some form of nullFlavor; other information may still be provided.

The context of use may also apply a cardinality to an attribute. Except for attributes with the type COLL and it's specializations, the only cardinalities that may be applied are 0..0, 0..1, and 1..1. Cardinality of 0 means that the attribute is not be represented in the instance, and has an implicit nullFlavor of NI. Cardinality of 1 means that the attribute has a value, though

© All rights reserved

the value may be a nullFlavor unless the attribute is also flagged as mandatory. For attributes with a collection type, the cardinality specifies how many items are in the collection. A cardinality limit of * means that there is no limit to the number of items in the collection. A mandatory attribute SHALL have a minimum cardinality of 1.

Note: this use of cardinality is a little different than the standard use of cardinality on attributes in UML. In UML, if an attribute is assigned a type of SET(CS) and a cardinality of 2..3, this means that there must be 2 or 3 sets of CS. These uses are not incompatible; both forms of cardinality may be applied in any information processing entity that claims direct or indirect conformance. Which form is intended should be made clear in the documentation.

In the case of a mandatory collection, the collection SHALL not contain any null items.

7.2 Top Level Model

For convenience and reference, The following diagram provides an overview of the datatypes defined in this specification as a UML diagram.

© All rights reserved

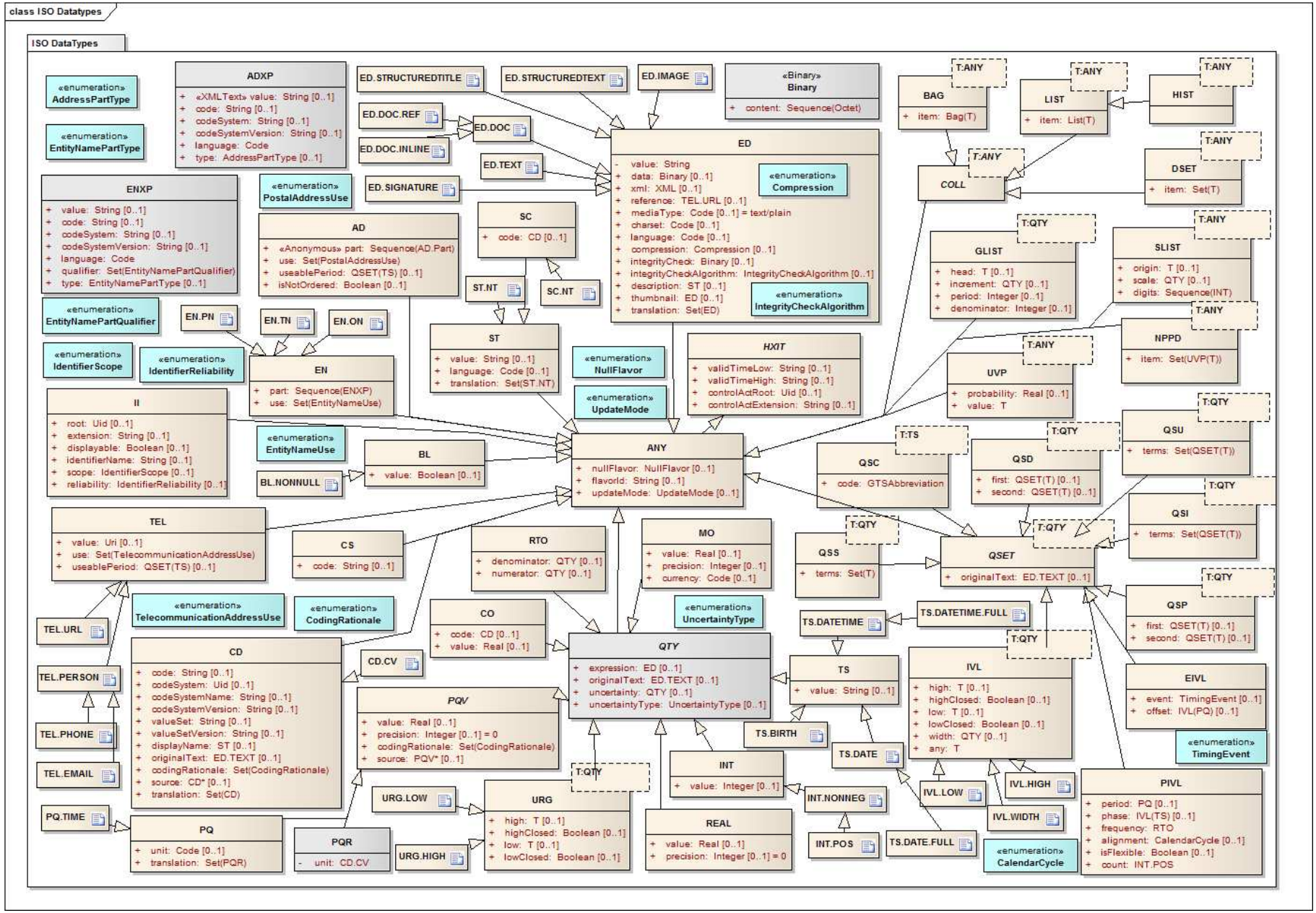


Figure 1. Top Level Model

© All rights reserved

7.3 Basic Data Types

7.3.1 Overview

Basic data types that provide infrastructural support for specific data types that are defined in subsequent sections.

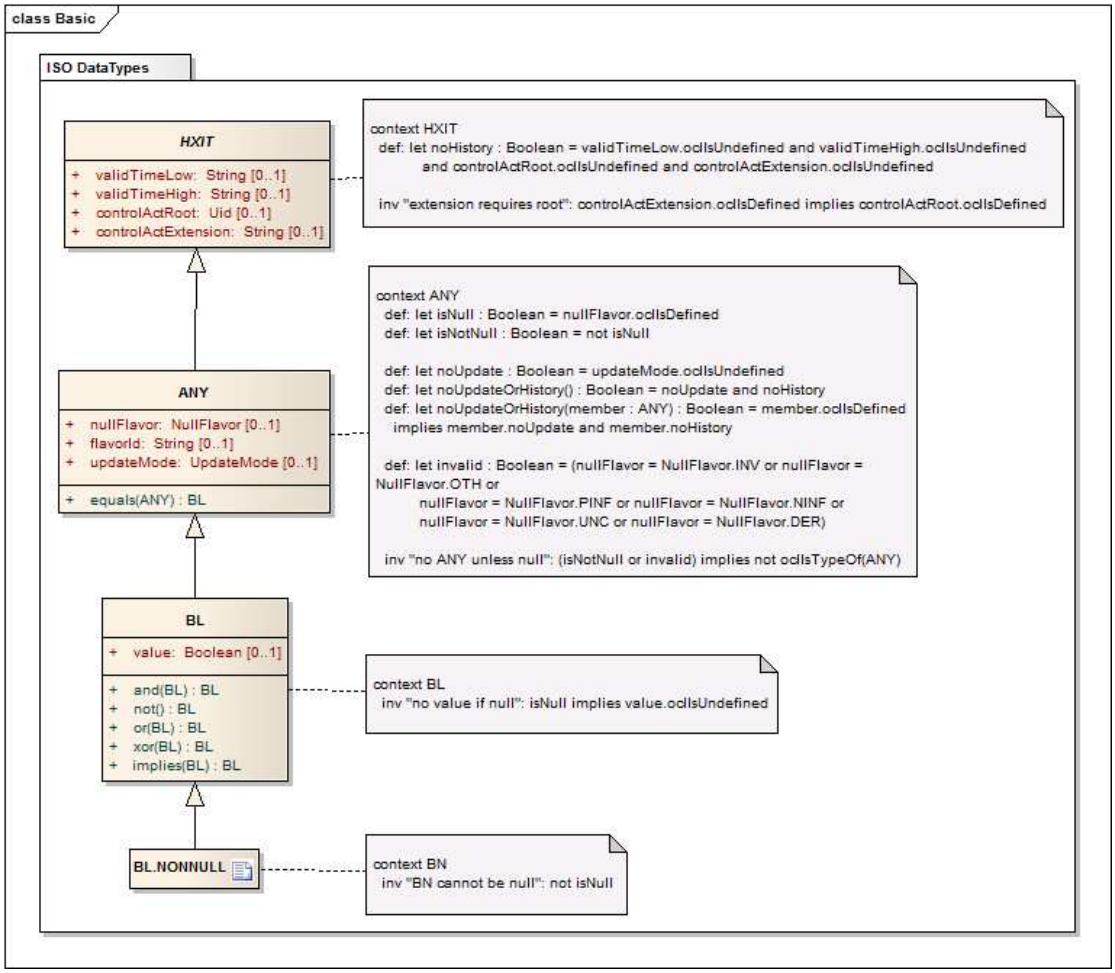


Figure 2. Basic Data Types

HXIT

7.3.2.1 Description

Abstract and Private –this datatype is not for use outside the data types in this specification.

Information about the history of this value: period of validity and a reference to an identified event that established this value as valid.

Notes:

1. Because of the way that the types are defined, a number of attributes of the data types have values with a type derived from HXIT. In these cases the HXIT attributes are constrained to null. The only case where the HXIT attributes are allowed within a data type is on items in a collection (DSET, LIST, BAG, HIST).

© ISO 2001 – All rights reserved

2. The use of these attributes is generally subject to further constraints in the specifications make use of these types.

7.3.2.2 ISO 11404 Syntax

```
type HXIT = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring
)
```

7.3.2.3 Attributes

7.3.2.3.1 validTimeLow : String: The time that the given information became or will become valid.

Note: This is not the time that any system first observed the value, but the time that the actual value became correct (i.e. when a patient changes their name).

7.3.2.3.2 validTimeHigh : String: The time that the given information ceased or will cease to be correct.

Note: Both validTimeLow and validTimeHigh must be valid timestamps using the format described in 7.8.13.3.1 (TS.value).

7.3.2.3.3 controlActRoot : Uid: The root of the identifier of the event associated with setting the data type to its specified value.

Note: By referencing a particular event, the property links to all of the information surrounding that event, such as who made the change, when it was made, why it was made, what system originated the change, etc, as specified in the implementation specification.

7.3.2.3.4 controlActExtension : String: The extension of the identifier of the event associated with setting the data type to its specified value.

7.3.2.4 Equality

There is no equality definition for HXIT, since it is an abstract and private type. The attributes of HXIT (validTimeLow, validTimeHigh, controlActRoot, controlActExtension) never participate in the determination of equality of specializations of HXIT.

7.3.2.5 Invariants

If a controlActExtension is provided, a ControlActRoot must also be provided

OCL for invariants:

```
def: let noHistory : Boolean =
     validTimeLow.oclIsUndefined and
     validTimeHigh.oclIsUndefined and
     controlActRoot.oclIsUndefined and
     controlActExtension.oclIsUndefined
```


© ISO 2007 – All rights reserved

```
inv "extension requires root":
      controlActExtension.oclIsDefined implies
      controlActRoot.oclIsDefined
```

7.3.2.6 Example

```
<example
  xsi:type="ST" value="This is some content"
  validTimeLow="200506011000" validTimeHigh="200507031500"
    controlActRoot="1.2.3.4.5.6">
</example>
```

In this example, the value of the example attribute "This is some content" was valid from the 1st June 2005 10:00am to 3rd July 2005 3:00 pm. The value was set to "This is some content" by an event which is identified by the OID of 1.2.3.4.5.6.

7.3.3 ANY

7.3.3.1 Description

Specializes HXIT

Defines the basic properties of every data value. This is conceptually an abstract type, meaning that no proper value can be just a data value without belonging to any concrete type. Every public concrete type is a specialization of this general abstract DataValue type.

However exceptional values (null values) may be of type ANY, except for the exceptional values that imply the nullFlavor INV, since this requires a type to be meaningful.

7.3.3.2 ISO 11404 Syntax

```
type ANY = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring
)
```

Note: The appropriate use of all three of the attributes on ANY is intimately bound to the specification with which the data types are used, and generally that specification will need to establish special ways to control their use. Information Process Entities claiming direct or indirect conformance SHALL make it clear how the use of these three attributes is controlled.

7.3.3.3 Attributes

7.3.3.3.1 **nullFlavor : NullFlavor**: If the value is null, indicates the reason.

Though the nullFlavor concept has some relationship with the UML/OCL null, it is not the same thing, and the relationship and differences between the two must be understood to

© ISO 2007 – All rights reserved

properly implement this specification. For further discussion, see section 7.1.4 (Null and NullFlavor). Note that nullFlavor includes the concept of a UML null value, and also includes potentially fully populated instances that do not conform to the requirements placed on the instance (also known as "exceptional instances“). Both nonNull and null values SHALL always be valid according to the rules expressed in this specification.

If populated, the value of this attribute SHALL be taken from the HL7 NullFlavor code system. The current values are:

NullFlavor Enumeration. OID: 2.16.840.1.113883.5.1008			
1	NI	no information	The value is exceptional (missing, omitted, incomplete, improper). No information as to the reason for being an exceptional value is provided. This is the most general exceptional value. It is also the default exceptional value
2	INV	Invalid	The value as represented in the instance is not a member of the set of permitted data values in the constrained value domain of a variable
3	OTH	Other	The actual value is not a member of the set of permitted data values in the constrained value domain of a variable. (e.g., concept not provided by required code system)
4	PINF	positive infinity	Positive infinity of numbers
4	NINF	negative infinity	Negative infinity of numbers
3	UNC	unencoded	No attempt has been made to encode the information correctly but the raw source information is represented (usually in originalText)
3	DER	derived	An actual value may exist, but it must be derived from the provided information (usually an expression is provided directly)
2	UNK	unknown	A proper value is applicable, but not known
3	ASKU	asked but unknown	Information was sought but not found (e.g., patient was asked but didn’t know)
4	NAV	temporarily unavailable	Information is not available at this time but it is expected that it will be available later
3	NASK	not asked	This information has not been sought (e.g., patient was not asked)
3	QS	sufficient quantity	The specific quantity is not known, but is known to be non-zero and is not specified because it makes up the bulk of the material.'Add 10mg of ingredient X, 50mg of ingredient Y, and sufficient quantity of water to 100mL.' The null flavor would be used to express the quantity of water
3	TRC	trace	The content is greater than zero, but too small to be quantified
2	MSK	masked	There is information on this item available but it has not been provided by the sender due to security, privacy or other reasons. There may be an alternate mechanism for gaining access to this information.

© ISO 2007 – All rights reserved

			Warning: Using this null flavor does provide information that may be a breach of confidentiality, even though no detail data is provided. Its primary purpose is for those circumstances where it is necessary to inform the receiver that the information does exist without providing any detail
2	NA	not applicable	No proper value is applicable in this context (e.g., last menstrual period for a male)

11404 Syntax for nullFlavor attribute

```
 type NullFlavor = enumerated (NI, INV, OTH, NINF, PINF, UNC,
DER, UNK, ASKU, NAV, QS, NASK, TRC, MSK, NA)
```

Some of the null flavors are not generally applicable to all data types. The nullFlavors NINF, PINF, QS, and TRC SHALL only be used in associated with QTY types. The nullFlavor UNC SHALL only be used with any type that has an originalText, and when UNC is used the originalText property SHALL be populated. When the nullFlavor DER is used, an expression SHALL be provided.

7.3.3.3.2 updateMode : UpdateMode: This property allows a sending system to identify the role that the attribute plays in processing the instance that is being represented.

If populated, the value of this attribute SHALL be taken from the HL7 UpdateMode code system. The current values are:

UpdateMode Enumeration. OID: 2.16.840.1.113883.5.57			
1	A	add	The item was (or is to be) added, having not been present immediately before. (If it is already present, this may be treated as an error condition.)
1	D	delete	Delete any previously received value. If the item is part of a collection, delete any matching items
1	R	replace	The item existed previously and has (or is to be) revised. (If an item does not already exist, this may be treated as an error condition.)
1	AR	add or replace	The item was (or is to be) either added or replaced.
	N	No Change	There was (or is to be) no change to the item. This is primarily used when this element has not changed, but other attributes in the instance have changed.
1	U	Unknown	It is not specified whether or what kind of change has occurred to the item, or whether the item is present as a reference or identifying property.
1	K	key	This item is part of the identifying information for this object

© ISO 2007 – All rights reserved

11404 Syntax for updateMode attribute

```
type UpdateMode = enumerated (A, D, R, AR, N, U, K)
```

If no updateMode is provided, there is no information as to how this information updates any existing information. The descriptions above use the word “matching”. For the purposes of data types, this means the equality operations defined in this specification (in other contexts “matching” may have other meanings).

> Note that updateMode does not affect the semantics or behaviour of the data type itself, but may affect the behaviour of systems processing objects containing instances of the data type.

7.3.3.3.3 flavorId : String: Signals the imposition of a set of constraints on the data type. The sole purpose of specifying that a constraint that has been used to further constrain the data type is to support validation of the instance: a validation engine can look up the rules expressed for the type and confirm that the instance conforms to the rules for the flavor. No other processing should depend on the content of the flavor attribute**.**

No other semantic or computational use SHALL depend on the value of this property. If this value is populated, the data type flavor SHALL be a valid constraint on the type of the value. f this value is populated, the data type flavor SHALL be a valid constraint on the type of the value.

7.3.3.4 Equality

By default, equals is determined as specified below. Selected specializations of ANY override equals to specify how semantic equality is evaluated for the type. Each type clearly documents how equality is determined.

The following table summarises the relationship between null, nullFlavor, and equals:

other **this**	null	nullFlavor	proper value
Null	null	null	Null
nullFlavor	null	nullFlavor‡	nullFlavor‡
proper value	null	nullFlavor‡	proper value†

‡ first common generalization of both nullFlavors
† unless specifically defined for a specialization, use the general equality algorithm.

Notes:

- The general equality algorithm says that two values are equal if they have the same type, and if all the attributes not defined on HXIT or ANY are also equal. If any of the attribute‘s equality is null or a nullFlavor then the result is null or has the most common nullFlavor. Note the comments on comparing nullValues in 7.1.4 (Null and NullFlavors).

- The equality operation is *reflexive*, *symmetric*, *transitive*, and *consistent*, and implementations must conform to these requirements. The equality rules defined in this specification conform to these requirements.

- This operation conforms to the general rules for operations and nullFlavors defined in Section 7.1.4 (Null and NullFlavor), but the rules are described here in depth for greater clarity. In particular, the rules are transitive within the equals operation: if any of the attributes or

© ISO 2007 – All rights reserved

collection items are null (other than the nullFlavor NA), the result will become null of the appropriate nullFlavor, unless specifically defined otherwise.

- equals does not override the = operation defined in OCL, nor should it override the normal equivalent for the OCL = operation in any implementation platform. It defines semantic equality.

- updateMode and flavorId are always ignored when testing for equality.

7.3.3.5 Invariants

- A instance may only be of type ANY (not a specialization) if it is null, and not if the nullFlavor implies INV

OCL for invariants:

```
def: let isNull : Boolean = nullFlavor.oclIsDefined
def: let isNotNull : Boolean = not isNull
def: let noUpdate : Boolean = updateMode.oclIsUndefined
def: let noUpdateOrHistory() : Boolean = noUpdate and
     noHistory
def: let noUpdateOrHistory(member : ANY) : Boolean =
     member.oclIsDefined implies member.noUpdate and
     member.noHistory

inv "no ANY unless null": (isNotNull or
   nullFlavor.implies(NullFlavor.INV)) implies
   not oclIsTypeOf(ANY)
```

7.3.3.6 Operations

7.3.3.6.1 equals(other : ANY) : BL: Defines whether the this data value is considered semantically equal to another data value - that they carry the same meaning. See the equality section above () for a discussion on how equality is determined.

7.3.3.6.2 isNull() : Boolean: Defines whether this type has a nullFlavor or not.

Note: This operation is an exception from the normal rules for operations and nullFlavors: it will return true if the value has a nullFlavor, and false if it does not.

7.3.3.6.3 notNull() : Boolean: Defines whether this type doesn't have a nullFlavor or it does.

Note: This operation is an exception from the normal rules for operations and nullFlavors: it will return false if the value has a nullFlavor, and true if it does not.

© ISO 2007 – All rights reserved

7.3.4 BL

7.3.4.1 Description:

Specializes ANY

BL stands for the values of two-valued logic. A BL value can be either true or false, or may have a nullFlavor.

7.3.4.2 ISO 11404 Syntax:

```
type BL = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  value : boolean
)
```

With any data value potentially being NULL, the two-valued logic is effectively extended to a three-valued logic as shown in the following truth tables:

NOT			AND	true	false	null		OR	true	false	null
True	false		true	true	false	null		true	true	true	True
false	true		false	false	false	false		false	true	false	null
Null	null		null	null	false	null		null	true	null	null

In this table, null stands for either true null or a nullFlavor. If the null (or either of the nulls) is a true null, the result will be a true null. Where a boolean operation is performed upon 2 datatypes with different nullFlavors, the nullFlavor of the result is the first common generalization of the 2 different nullFlavors, though conformant applications may also create a result that is any common generalization (such as NI).

7.3.4.3 Attributes

7.3.4.3.1 value: Boolean: The value of the BL if it is non Null.

7.3.4.4 Equality

Two boolean values are equal if they are nonNull and have the same value.

7.3.4.5 Invariants

The BL must have a value if it is not null

© ISO 2007 – All rights reserved

OCL for invariants:

```
inv "no value if null": isNotNull implies value.oclIsDefined
```

7.3.4.6 Operations

7.3.4.6.1 and(other : BL) : BL: True if both values are true. False if either value is false. Null otherwise.

7.3.4.6.2 or(other : BL) : BL: True if either this or other are true. False if both values are false. Null Otherwise

7.3.4.6.3 xor(other : BL) : BL: True if either this and other are different and not null. False if both this and other have the same value. Null otherwise.

7.3.4.6.4 implies(other : BL) : BL: True if either this is false, or if this and other are true. False if this this is true and other is false. Null otherwise.

7.3.4.6.5 not() : BL: False if this true, true if this is false. Null otherwise.

Note: These operations do not always confom to the general rules for nullFlavors and operations due to the special nature of these logical operations. For example. Null or True is true, because it does not matter whether the null value might actually be true or false, the result will be the same. Also note, from an implementers point of view, that these operations are strictly symmetric, and this may require som special case handling to prevent implementation errors when performing operations on null objects.

7.3.4.7 Examples

7.3.4.7.1 Simple True Value

```
<example xsi:type="BL" value="true"/>
```

7.3.4.7.2 Unknown Value

```
<example xsi:type="BL" nullFlavor="UNK"/>
```

7.3.5 BL.NONNULL

7.3.5.1 Description:

Constrains BL

BL.NONNULL is a constrained instance of BL that cannot have a nullFlavor.

7.3.5.2 Invariants

- BL.NONNULL cannot have a nullFlavor

© ISO 2007 – All rights reserved

OCL for invariants:

```
inv "cannot be null": not isNull
```

7.4 Text And Binary Data Types

© ISO 2007 – All rights reserved

7.4.1 Overview

Data types that provide support for text and multimedia data.

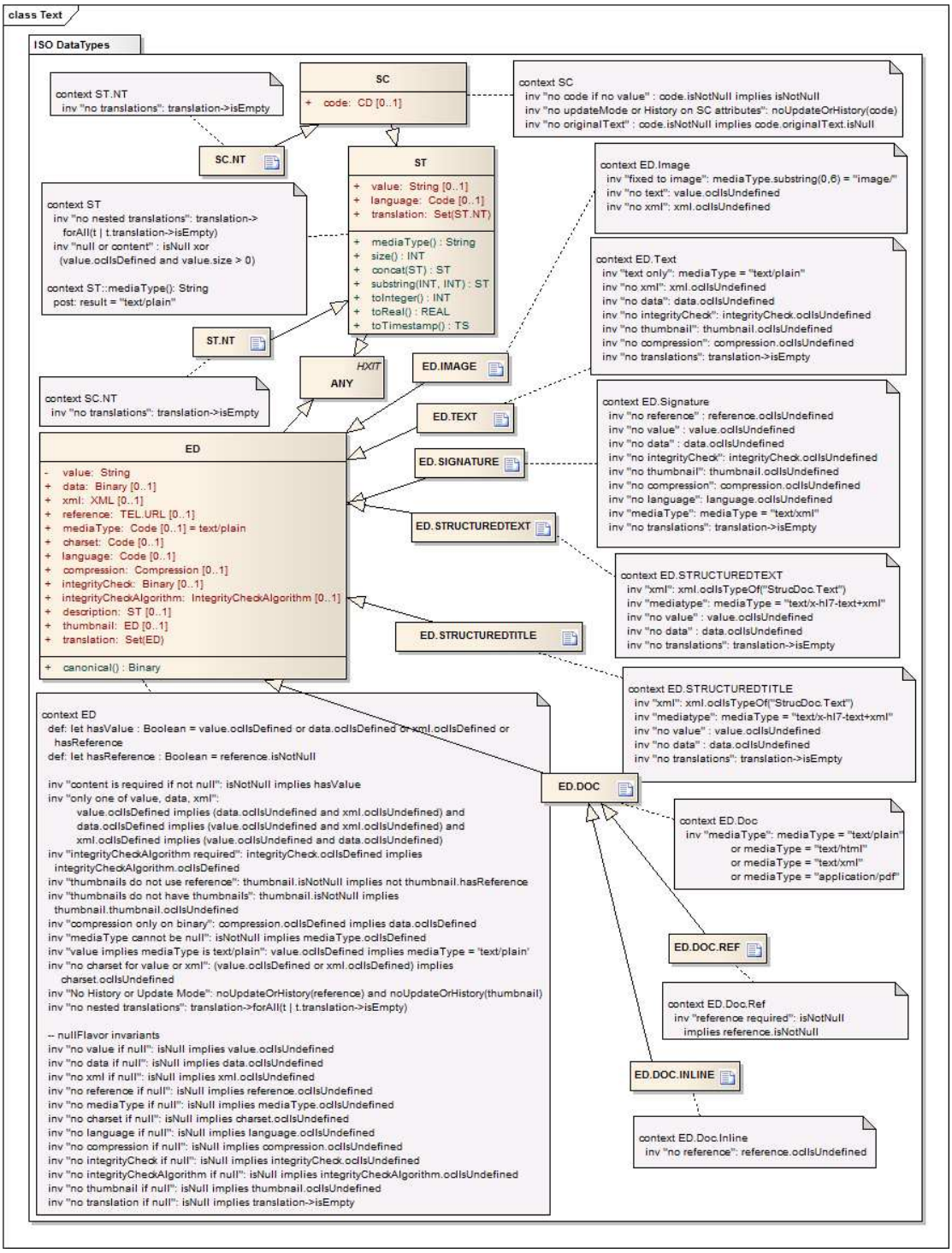


Figure 3. Text and binary data types

© ISO 2007 – All rights reserved

7.4.2 ED (Encapsulated Data)

7.4.2.1 Description:

Specializes ANY

Data that is primarily intended for human interpretation or for further machine processing outside the scope of this specification. This includes unformatted or formatted written language, multimedia data, or structured information as defined by a different standard (e.g., XML-signatures.)

Note 1: Encapsulated data can be present in two forms, inline or by reference. Inline data is communicated or moved as part of the encapsulated data value, whereas by-reference data may reside at a different (remote) location. The content is the same whether it is located inline or remote.

Note 2: The actual content - a sequence of bytes (which corresponds to the type BIN defined in the HL7 V3 abstract datatypes) - may be provided in one of 4 different ways:

- as a plain sequence of characters (value)
- as a binary (a sequence of bytes) (data)
- as xml content (xml)
- as a URL/URI that provides reference to the information required to locate the data (reference)

Content SHALL be provided if the ED is not null. The content may be provided in-line (using only one of value, data or xml), or it may be provided as a reference.Content may be provided in-line and a reference also may be given; in these cases, it is expected that the content of the reference will be exactly the same as the in-line content. Information Processing Entities are not required to check this, but may regard it as an error condition if the content does not match.

7.4.2.2 ISO 11404 Syntax:

```
type ED = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  value : characterstring,
  data : Sequence(Octet),
  xml : XML,
  reference : TEL.URL,
  mediaType : characterstring,
  charset : characterstring,
  language : characterstring,
  compression : Compression,
  integrityCheck : Sequence(Octet),
  integrityCheckAlgorithm : IntegrityCheckAlgorithm,
  description : characterstring,
  thumbnail : ED,
  translation : SET(ED)
```


© ISO 2007 – All rights reserved

)

7.4.2.3 Attributes

7.4.2.3.1 value : String: A simple sequence of characters that contains the content.

Note: If value is used, the mediatype is fixed to text/plain and the charset must be consistent with the String Character Set. Refer to section 6.7.4 for more details.

7.4.2.3.2 data : Binary: A simple sequence of byte values that contains the content.

7.4.2.3.3 xml : XML: The content represented in plain XML form.

Note: a direct representation is provided for XML. This is because this specification includes an XML serialization of the data, and this xml attribute is handled specially in the serialisation form. The xml data is not different in any semantic sense to the same data if represented in the value or data attributes.

Note: These three representations of the ED data –as a sequence of characters, as a sequence of bytes, or as XML in a native XML format – are mutually incompatible and could also have been implemented as 3 specialisations of an abstract ED supertype. However doing so would complicate definition and implementation of the ED flavors and complicate the associated XML format (the additional of a mandatory xsi:type attribute) without significantly simplifying the overall implementation of ED.

7.4.2.3.4 reference : TEL.URL : A telecommunication address such as a URL for HTTP or FTP, which will resolve to precisely the same binary content that could as well have been provided as inline content.

Note: The semantic value of an encapsulated data value is the same, regardless whether the content is present as inline content or just by reference. However, an encapsulated data value without inline content behaves differently, since any attempt to examine the content requires the data to be downloaded from the reference. An encapsulated data value may have both inline content and a reference.

If data is provded in the value, data or xml attributes, the reference SHALL point to the same data. It is an error if the data resolved through the reference does not match either the integrity check, data as provided, or data that had earlier been retrieved through the reference and then cached.

The reference may contain a usablePeriod to indicate that the data may only be available for a limited period of time. Whether the reference is limited by a usablePeriod or not, the content of the reference SHALL be fixed for all time. Any application using the reference SHALL always receive the same data, or an error. The reference cannot be reused to send a different version of the same data, or different data.

7.4.2.3.5 mediaType : Code: Identifies the type of the encapsulated data and identifies a method to interpret or render the content.

The IANA defined domain of media types is established by the IETF RFCs 2045 and 2046. mediaType has a default value of text/plain and cannot be null.

mediaType is a mandatory property, i.e., every non-NULL instance of *ED* SHALL have a non-NULL *mediaType* property.

© ISO 2007 – All rights reserved

7.4.2.3.6 charset : Code: For character-based encoding types, this attribute specifies the character set and character encoding used in the attribute that contains the content of the ED.

The charset SHALL be identified by an Internet Assigned Numbers Authority (IANA) Charset Registration in accordance with IETF RFC 2978.

Note: Whenever the content of the ED is character type data in any form, the charset property needs to be known. If the content is provided directly in the value attribute, then the charset SHALL be a known character set consistent with the String Character Set. Refer to section 6.7.4 for more details. If the content is provided as a reference, and the access method does not provide the charset for the content (such as by a mime header), then the charset SHALL be conveyed as part of the ED.

7.4.2.3.7 language: Code: The human language of the content. Valid codes are taken from the IETF RFC 3066. If this attribute is null, the language may be inferred from elsewhere, either from the context or from unicode language tags, for example.

Note: While language attribute usually alters the interpretation of the text, the language attribute does not alter the meaning of the characters in the text.

7.4.2.3.8 compression : Compression: Indicates whether the raw byte data is compressed, and what compression algorithm was used.

If the attribute is null, the data is not compressed. Compression only applies to the binary form of the content.

If populated, the value of this attribute SHALL be taken from the HL7 CompressionAlgorithm code system. The current values are:

CompressionAlgorithm Enumeration. OID: 2.16.840.1.113883.5.1009			
1	DF	deflate	The deflate compressed data format as specified in IETF RFC 1951
1	GZ	gzip	A compressed data format that is compatible with the widely used GZIP utility as specified in IETF RFC 1952(uses the deflate algorithm)
1	ZL	zlib	A compressed data format that also uses the deflate algorithm. Specified as IETF RFC 1950
1	Z	compress	Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than deflate
1	BZ	bzip	bzip-2 compression format. See [http://www.bzip.org/] for more information.
1	Z7	Z7	7z compression file format. See [http://en.wikipedia.org/wiki/7z] for more information.

Note that some compression formats allow multiple archive files to be embedded within a single compressed volume. Applications SHALL ensure that the decompressed form of the data conforms to the stated media type.

11404 Syntax for compression attribute

```
type Compression = enumeration (DF, GZ, ZL, Z, BZ, Z7)
```


© ISO 2007 – All rights reserved

7.4.2.3.9 integrityCheck : Binary: The integrity check is a short binary value representing a cryptographically strong checksum that is calculated over the binary content.

Note: The purpose of this property, when communicated with a reference is for anyone to validate later whether the reference still resolved to the same content that the reference resolved to when the encapsulated data value with reference was created. If the attribute is null, there is no integrityCheck.

It is an error if the data resolved through the reference does not match the integrity check.

The integrity check is calculated according to the integrityCheckAlgorithm. By default, the Secure Hash Algorithm-1 (SHA-1) shall be used. The integrity check is binary encoded according to the rules of the integrity check algorithm.

The integrity check is calculated over the raw binary data that is contained in the data component, or that is accessible through the reference. No transformations are made before the integrity check is calculated. If the data is compressed, the Integrity Check is calculated over the compressed data.

7.4.2.3.10 integrityCheckAlgorithm : IntegrityCheckAlgorithm: Specifies the algorithm used to compute the integrityCheck value.

If populated, the value of this attribute SHALL be taken from the HL7 IntegrityCheckAlgorithm code system. The current values are:

IntegrityCheckAlgorithm Enumeration. OID: 2.16.840.1.113883.5.1010			
1	SHA1	secure hash algorithm - 1	This algorithm is defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995
1	SHA256	secure hash algorithm - 256	This algorithm is defined in FIPS PUB 180-2: Secure Hash Standard

11404 Syntax for integrityCheckAlgorithm attribute

```
type IntegrityCheckAlgorithm = enumeration (SHA1, SHA256)
```

7.4.2.3.11 description : ST: A stand-in for the full media for use as an alternative description of the media where the media is not able to be rendered.

E.g. Short text description of an image or sound clip, etc. This attribute is not intended to be a complete substitute for the original. For complete substitutes, use the "translation" property.

Note: The intent of this property is to allow compliance with disability requirements such as those expressed in American's with Disability Act (also known as "Section 508"), where there is a requirement to provide a short text description of included media in some form that can be read by a screen reader.

7.4.2.3.12 thumbnail : ED: An abbreviated rendition of the full content.

Note: A thumbnail requires significantly fewer resources than the full content, while still maintaining some distinctive similarity with the full content. A thumbnail is typically used with by-reference encapsulated data. It allows a user to select the appropriate content more efficiently before actually downloading through the reference.

Originally, the term thumbnail refers to an image in a lower resolution (or smaller size) than another image. However, the thumbnail concept can be metaphorically used for media types other than images. For example, a movie may be represented by a shorter clip; an audio-clip may be represented by another audio-clip that is shorter, has a lower sampling rate, or a lossy compression; or an abstract provided for a long document.

© ISO 2007 – All rights reserved

A thumbnail itself SHALL NOT contain a thumbnail.

7.4.2.3.13 translation : Set(ED): Alternate renditions of the same content translated into a different language or a different mediaType.

The translations SHALL convey the same information, but in a different language or mediaType. Translations SHALL not contain translations. The translations do not take part in the test for equality, so SHALL NOT introduce any new semantics to the value.

7.4.2.4 Equality

Two nonNull values of type *ED* are equal if and only if their mediaType and data are equal. For those *ED* values with compressed data or referenced data, only the de-referenced and uncompressed data counts for the equality test (the canonical content, see section 7.4.2.6.1). The compression, thumbnail, translation and reference property themselves are excluded from the equality test. In addition the language property is excluded from the test (refer to the note for the language attribute). If the mediaType is character based and the charset property is not equal, the charset property must be resolved through mapping of the data between the different character sets.

An ED with a plain text content may also be equal to a ST with the same character content, following the rules described above.

7.4.2.5 Invariants

- Either reference, data, value or xml must be provided if not null
- Only one of reference, data, value or xml may be specified
- An integrityCheckAlgorithm must be provided if an integrityCheck is provided
- if a thumbnail is provided, it must not use a reference
- if a thumbnail is provided, it must not have a thumbnail
- Compression can only be specified if data is provided as a binary
- mediaType cannot be null
- if value is used, the mediaType is plain text
- a character set must not be asserted for plain text or xml content (for plain text refer to 6.7.4, and implicitly derived for the XML content)
- translations may not contain translations

OCL for invariants:

```
def: let hasReference : Boolean = reference.isNotNull
def: let hasValue : Boolean = value.oclIsDefined or
      data.oclIsDefined or xml.oclIsDefined or hasReference
inv "content is required if not null": isNotNull implies
     hasValue
inv "only one of value, data, xml": value.oclIsDefined
```


© ISO 2007 – All rights reserved

```
        implies (data.oclIsUndefined and xml.oclIsUndefined)
        and data.oclIsDefined implies (value.oclIsUndefined and
        xml.oclIsUndefined) and xml.oclIsDefined implies
       (value.oclIsUndefined and data.oclIsUndefined)
   inv "integrityCheckAlgorithm
        required":integrityCheck.oclIsDefined implies
        integrityCheckAlgorithm.oclIsDefined
   inv "thumbnails do not use reference": thumbnail.isNotNull
       implies not thumbnail.hasReference
   inv "thumbnails do not have thumbnails": thumbnail.isNotNull
         implies thumbnail.thumbnail.oclIsUndefined
   inv "compression only on binary": compression.oclIsDefined
         implies data.oclIsDefined
   inv "mediaType cannot be null": isNotNull implies
         mediaType.oclIsDefined
   inv "value implies mediaType is text/plain":
         value.oclIsDefined implies mediaType = 'text/plain'
   inv "no charset for value or xml": (value.oclIsDefined or
         xml.oclIsDefined)implies charset.oclIsUndefined
   inv "No History or Update Mode":noUpdateOrHistory(reference)
         and noUpdateOrHistory(thumbnail)
   inv "no nested translations": translation->forAll(t |
       t.translation->isEmpty)
```

nullFlavor invariants

```
     inv "no value if null": isNull implies
          value.oclIsUndefined
     inv "no data if null": isNull implies data.oclIsUndefined
     inv "no xml if null": isNull implies xml.oclIsUndefined
     inv "no reference if null": isNull implies
         reference.oclIsUndefined
     inv "no mediaType if null": isNull implies
         mediaType.oclIsUndefined
     inv "no charset if null": isNull implies
          charset.oclIsUndefined
     inv "no language if null": isNull implies
         language.oclIsUndefined
     inv "no compression if null": isNull implies
          compression.oclIsUndefined
     inv "no integrityCheck if null": isNull implies
          integrityCheck.oclIsUndefined
     inv "no integrityCheckAlgorithm if null": isNull
         implies integrityCheckAlgorithm.oclIsUndefined
     inv "no thumbnail if null": isNull implies
         thumbnail.oclIsUndefined
     inv "no translation if null": isNull implies translation->
          isEmpty
```

7.4.2.6 Operations

7.4.2.6.1 canonical : Binary: The sequence of bytes that is the actual data.

Notes:

© ISO 2007 – All rights reserved

1. This sequence of bytes is retrieved from the reference if one was provided, and decompressed if appropriate.

2. This operation does not follow the normal rules for operations and nullFlavors because the return type cannot have a nullFlavor.

3. The result is null if the ED has a nullFlavor.

7.4.2.7 Examples

7.4.2.7.1 Plain Text

```
<example xsi:type="ED" value="this is plain text"
    mediaType="text/plain"/>
<example xsi:type="ED" value="this is plain text"/>
```

This ED is a simple representation of plain text. The mediaType is specified as text/plain. Since the default value for mediatype is text/plain, the mediaType does not need to be represented in the XML, and the second example is also valid. The character set is not specified, and in this case, the character set must not be present; the character set matches the encoding of the XML.

7.4.2.7.2 Language

```
<example xsi:type="ED" flavorId="ED.TEXT" value="this is plain text"
language="en" mediaType="text/plain"/>
<example xsi:type="ED" flavorId="ED.TEXT" value="dieses ist normaler
Text" language="de" mediaType="text/plain"/>
```

This explicitly notes that the language is English in the first case and german in the second case. If there is no language attribute, then the language is unknown, though it is usually safe to assume that the locally predominant language is appropriate. The type is assigned the flavour ED.TEXT - a text only flavor of ED.

```
<example xsi:type="ED" value="this is plain text" language="en">
  <translation xsi:type="ED" value="dieses ist normaler Text"
    language="de"/>
</example>
```

Here the german version is contained as a translation of the english version.

7.4.2.7.3 Binary Content

```
<example xsi:type="ED">
  <data>dGhpcyBpcyBiaW5hcnkgY29udGVudA==</data>
</example>
```

This ED contains the plain text "this is binary content". Since the mediaType is still text plain, the mediaType attribute still doesn’t need to be populated. Note that the character set of this plain text is unknown; it does not necessarily match that of the XML, and it is not safe to assume that it does.

```

```


© ISO 2007 – All rights reserved

```
<example xsi:type="ED" charset="UTF-8">
  <data>dGhpcyBpcyBiaW5hcnkgY29udGVudA==</data>
</example>
```

For this reason it would generally be appropriate to define a charset when using the data element, as shown in this example.

7.4.2.7.4 XML Content

```
<example xsi:type="ED" mediaType="text/xml">
  <data>PHBhcmVudD4NCiAgPGNoaWxkPlRoaXMgaXMgc29tZSB0ZXh0IGluI
    HRoZSBjaGlsZDwvY2hpbGQ+
  DQogIFRoaXMgaXMgc29tZSB0ZXh0IGluIHRoZSBwYXJlbnQNCjwvcG
   FyZW50Pg==</data>
</example>
```

This ED contains some XML content provided in binary form. Like the previous example, the character set of the XML content is unknown; it does not necessarily match that of the ED XML, and it is not safe to assume that it does.

```
<example xsi:type="ED" mediaType="text/xml" charset="ASCII">
  <data>PHBhcmVudD4NCiAgPGNoaWxkPlRoaXMgaXMgc29tZSB0ZXh0IGlu
    IHRoZSBjaGlsZDwvY2hpbGQ+
  DQogIFRoaXMgaXMgc29tZSB0ZXh0IGluIHRoZSBwYXJ
    lbnQNCjwvcGFyZW50Pg==</data>
</example>
```

The charset is explicitly defined in this case. It does not need to match the XML document encoding.

```
<example xsi:type="ED" mediaType="text/xml">
  <xml>
    <parent>
      <child>This is some text in the child</child>
      This is some text in the parent
    </parent>
  </xml>
</example>
```

In this example, the xml is provided in-line using the xml element. The mediaType of the xml content must be provided (it cannot be text/plain). The charset cannot be defined since it must match that of the xml encoding.

7.4.2.7.5 Other Content Types

```
<example mediaType="application/pdf" compression="GZ">
   <data>/9j/4AAQSkZJRgABAgEAgACAAAD/2wCEAAICAgIC
      AQICAgICAgICAwQDAwMDAwUEBAMEBgYHBgYG
    BgYHCAoIBwcJBwYGCAsJCQoKCwsLBwgMDQwKDAoLCwoBAgIC
      AwMDBQMDBQoHBgcKCgoKCgoKCgoK
    CgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgo
      KCv/AABEIAg4CuQMBIQACEQED
    EQH/xAGiAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgsBAAM
      BAQEBAQEBAQEAAAAAAAABAgME
    BQYHCAkKCxAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1F
      hByJxFDKBkaEII0KxwRVS0fAk
    ...
    CigAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKAC
```

© ISO 2007 – All rights reserved

```
      gAooAKKACigAooAKKACigAoo
    AKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAK
      KACigAooAKKACigAooAKKACi
    gAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKACigA
      ooAKKACigAooAKKAP//Z</data>
</example>
```


This example shows an Adobe Acrobat document that has been compressed using the GZip compression algorithm.


```
<example mediaType="image/png">
   <reference value="http://example.org/xrays/128s8d9ej229se32s.png">
      <useablePeriod xsi:type="IVL_TS">
        <low value="200007200845"/>
        <high value="200008200845"/>
      </useablePeriod>
   </reference>
  <integrityCheck>EQH/xAGiAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgsB
     AAMBAQEBAQEBAQEAAAAAAAABAgME</integrityCheck>
   <thumbnail mediaType="image/jpeg">
      <data>/9j/4AAQSkZJRgABAgEAgACAAAD/2wCEAAICAgIC
      AQICAgICAgICAwQDAwMDAwUEBAMEBgYHBgYG
    BgYHCAoIBwcJBwYGCAsJCQoKCwsLBwgMDQwKDAoLCwoBAgIC
      AwMDBQMDBQoHBgcKCgoKCgoKCgoK
    CgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgo
      KCv/AABEIAg4CuQMBIQACEQED
    EQH/xAGiAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgsBAAM
      BAQEBAQEBAQEAAAAAAAABAgME
    BQYHCAkKCxAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1F
      hByJxFDKBkaEII0KxwRVS0fAk
    ...
    CigAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKAC
      gAooAKKACigAooAKKACigAoo
    AKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAK
      KACigAooAKKACigAooAKKACi
    gAooAKKACigAooAKKACigAooAKKACigAooAKKACigAooAKKACigA
      ooAKKACigAooAKKAP//Z </data>
   </thumbnail>
</example>
```


This example contains a reference to an image, stored at particular URL and available for the next month. An integrity check is provided for the image, as well as in inline thumbnail.

7.4.3 ED.IMAGE

7.4.3.1 Description

Constrains ED

ED.IMAGE constrains ED so that the contents must be an image.

7.4.3.2 Invariants

- The mediaType must start with “image/”
- The content cannot be provided as a text or xml – it must be binary and/or reference

© ISO 2007 – All rights reserved

OCL for invariants:

```
inv "fixed to image": mediaType.substring(0,6) = "image/"
inv "no text": value.oclIsUndefined
inv "no xml": xml.oclIsUndefined
```

7.4.4 ED.TEXT

7.4.4.1 Description

Constrains ED

ED.TEXT constrains ED so that it may only contain plain text.

Note: This is useful because there is sometimes a need to allow for references, but the content must be a simple string. In addition, no translations are allowed.

7.4.4.2 Invariants

- The mediaType must be text/plain
- The content cannot be provided as a text or data – it must be text and/or reference
- integrity check, thumbnail, compression and translations are not allowed

OCL for invariants:

```
inv "text only": mediaType = "text/plain"
inv "no xml": xml.oclIsUndefined
inv "no data": data.oclIsUndefined
inv "no integrityCheck": integrityCheck.oclIsUndefined
inv "no thumbnail": thumbnail.oclIsUndefined
inv "no compression": compression.oclIsUndefined
inv "no translations": translation->isEmpty
```

7.4.5 ED.DOC

7.4.5.1 Description

Constrains ED

ED.DOC constrains ED so that the contents must be a document or a reference to one.

7.4.5.2 Invariants

- mediaType.must be text/plain, text/html, text/xml or application/pdf

OCL for invariants:

```
inv "mediaType": mediaType = "text/plain" or mediaType =
    "text/html" or mediaType = "text/xml" or mediaType =
```

© ISO 2007 – All rights reserved

```
"application/pdf"
```

7.4.6 ED.DOC.INLINE

7.4.6.1 Description

Constrains ED.DOC

ED.DOC.INLINE constrains ED.DOC so that the document must not be provided by means of a reference.

7.4.6.2 Invariants

- No reference is allowed

OCL for invariants:

```
inv "no reference": reference.oclIsUndefined
```

7.4.7 ED.DOC.REF

7.4.7.1 Description

Constrains ED.DOC

ED.DOC.REF constrains ED.DOC so that the document must be provided by means of a reference.

7.4.7.2 Invariants

- A reference is required

OCL for invariants:

```
inv "reference required": isNotNull implies
      reference.isNotNull
```

7.4.8 ED.SIGNATURE

7.4.8.1 Description

Constrains ED

ED.SIGNATURE constrains ED so that the contents must be an XML digital signature.

7.4.8.2 Invariants

- No value, data, reference, integrity check, thumbnail, compression, language or translations are allowed

© ISO 2007 – All rights reserved

- The media type must be text/xml

OCL for invariants:

```
inv "no reference" : reference.oclIsUndefined
inv "no value" : value.oclIsUndefined
inv "no data" : data.oclIsUndefined
inv "no integrityCheck": integrityCheck.oclIsUndefined
inv "no thumbnail": thumbnail.oclIsUndefined
inv "no compression": compression.oclIsUndefined
inv "no language": language.oclIsUndefined
inv "mediaType": mediaType = "text/xml"
inv "no translations": translation->isEmpty
```

7.4.9 ED.STRUCTUREDTEXT

7.4.9.1 Description

Constrains ED

ED.STRUCTUREDTEXT constrains ED so that the contents must be a structured narrative.

See section 7.13 for the definition of a structured narrative.

7.4.9.2 Invariants

- The media type must be text/x-hl7-text+xml
- No value, data translations are allowed
- The xml must be a valid instance of the structured text defined in Section

OCL for invariants:

```
inv "mediatype": mediaType = "text/x-hl7-text+xml"
inv "no value" : value.oclIsUndefined
inv "no data" : data.oclIsUndefined
inv "no translations": translation->isEmpty
```

7.4.10 ED.STRUCTUREDTITLE

7.4.10.1 Description

Constrains ED

ED.STRUCTUREDTITLE constrains ED so that the contents must be a structured title.

See section 7.13 for the definition of a structured title.

© ISO 2007 – All rights reserved

7.4.10.2 Invariants

- The media type must be text/x-hl7-title+xml
- No value, data translations are allowed
- The xml must be a valid instance of the structured title defined in Section

OCL for invariants:

```
inv "mediatype": mediaType = "text/x-hl7-title+xml"
inv "no value" : value.oclIsUndefined
inv "no data" : data.oclIsUndefined
inv "no translations": translation->isEmpty
```

7.4.11 ST (Character String)

7.4.11.1 Description

Specializes ANY

The character string datatype stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.). Used for names, symbols, and formal expressions.

A ST SHALL have at least one character or else be NULL.

7.4.11.2 ISO 11404 Syntax:

```
type ST = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  value : characterstring,
  language : characterstring
  translation : SET(ST.NT)
)
```

7.4.11.3 Attributes

7.4.11.3.1 value : String: The actual content of the string. Refer to section 6.7.4 for discussion on String character encodings.

7.4.11.3.2 language: Code: The human language of the content. Valid codes are taken from the IETF RFC 3066. If this attribute is null, the language may be inferred from elsewhere, either from the context or from unicode language tags, for example.

Note: While Language tags usually alter the meaning of the text, the language does not alter the meaning of the characters in the text.

© ISO 2007 – All rights reserved

7.4.11.3.3 translation : Set(ST.NT): Alternate renditions of the same content translated into a different language. Translations may not contain translations.

The translations SHALL convey the same information, but in a different language. The translations do not take part in the test for equality, so SHALL NOT introduce any new semantics to the value.

7.4.11.4 Equality

Two nonNull values of type *ST* are equal if and only if the sequence of characters they represent are equal (i.e. if they are non-Null and their value attributes are equal). The translation property is excluded from the equality test. In addition the language property is excluded from the test, due to the problems this would introduce for values of type *ST* where the language is not specified.

An ED with a plain text content may also be equal to a ST with the same character content, following the rules described for ED (Section 0).

7.4.11.5 Invariants

- if there is a value, there must be at least one character
- translations may not contain translations

OCL for invariants:

```
inv "no nested translations": translation->
      forAll(t | t.translation->isEmpty)

inv "null or content" : isNull xor (value.oclIsDefined
      and value.size > 0)
```

7.4.11.6 Operations

7.4.11.6.1 mediaType : String: Returns a value of text/plain.

7.4.11.6.2 size : INT: The number of characters in the string.

7.4.11.6.3 concat(other : ST) : ST: The concatenation of this and other.

7.4.11.6.4 substring(lower : INT, upper : INT) : ST: The sub-string of this starting at character number lower, up to and including character number upper.

Character numbers run from 1 to this.size().

Note that when characters are extracted from a string, it may be necessary to copy other predecessor characters that set the appropriate context in some character encodings.

© ISO 2007 – All rights reserved

7.4.11.6.5 toInteger : INT: If the content of the string is a valid integer, the value as an INT. If the content is not a valid integer, then null (nullFlavor NI).

A string is a valid integer if it conforms to the integer-literal format defined in ISO 11404, or if it conforms to the lexical representation of the integer type defined in xml schema.

7.4.11.6.6 toReal : REAL: If the content of the string is a valid floating point number, the value as a REAL. If the content is not a valid floating point number, then null (nullFlavor NI).

A string is a valid floating point number if it conforms to the real-literal format defined in ISO 11404, or if it conforms to the lexical representation of the double type defined in xml schema.

7.4.11.6.7 toTimestamp : TS: If the content of the string is a valid timestamp, the value as a TS. If the content is not a valid timestamp, then null (nullFlavor NI). A string is a valid integer if it conforms to the format described under the TS type.

7.4.11.7 Examples

```
<example language="en" value="cellulitis of the left foot"/>
```

7.4.12 ST.NT

7.4.12.1 Description

Constrains ST

ST.NT constrains ST so that it no translations are allowed.

7.4.12.2 Invariants

- No translations are allowed

OCL for invariants:

```
inv "no translations": translation->isEmpty
```

7.4.13 SC (Coded String)

7.4.13.1 Description

Specializes ST

A character string that optionally may have a code attached. The text must always be present if a code is present.

Note: The code is often a local code. SC is used in cases where coding is exceptional (e.g., user error messages are essentially text messages, and the text message is the important content. However sometimes messages come from a catalog of prepared messages, which SC allows to reference).

© ISO 2007 – All rights reserved

Any non-null SC value MAY have a code, however, a code SHALL NOT be given without the text.

The similarities and differences between SC and CD are discussed in Section 7.5.1.2, CD and SC.

7.4.13.2 ISO 11404 Syntax:

```
type SC = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  value : characterstring,
  language : characterstring,
  translation : SET(ST),
  code : CD
)
```

7.4.13.3 Attributes

7.4.13.4 code : CD: The coded value associated with the string. If the value is null, there is no coded value associated with this string.

7.4.13.5 Equality

The definition of equality for SC is the same as for ST. Code is excluded from the equality test, and values of type SC may be equal to values to ST (and therefore to value of ED following the rules documented in for ST (Section 7.4.11.4).

7.4.13.6 Invariants

- If there is a code, there must also be some content on the SC (and therefore the SC must not be null)
- The originalText value of the CD must be null (The originalText is the SC.value)

OCL for invariants:

```
inv "no code if no value" : code.isNotNull implies
   isNotNull
inv "no updateMode or History on SC attributes":
    noUpdateOrHistory(code)
inv "no originalText" : code.isNotNull implies
   code.originalText.isNull
```

7.4.13.7 Examples

```
<example xsi:type="SC" language="en" value="Australia">
  <code code="AUS" codeSystem="2.16.840.1.113883.11.171"
codeSystemName="ISO 3166 / 3">
    <displayName value="Australia"/>
  </code>
```

© ISO 2007 – All rights reserved

```
</example>
```

7.4.14 SC.NT

7.4.14.1 Description

Constrains SC

SC.NT constrains SC so that it no translations are allowed.

7.4.14.2 Invariants

- No translations are allowed

OCL for invariants:

```
inv "no translations": translation->isEmpty
```


© ISO 2007 – All rights reserved

7.5 Coded Datatypes (Terminology)

These data types provide support for use of codes and terms from terminologies and classifications.

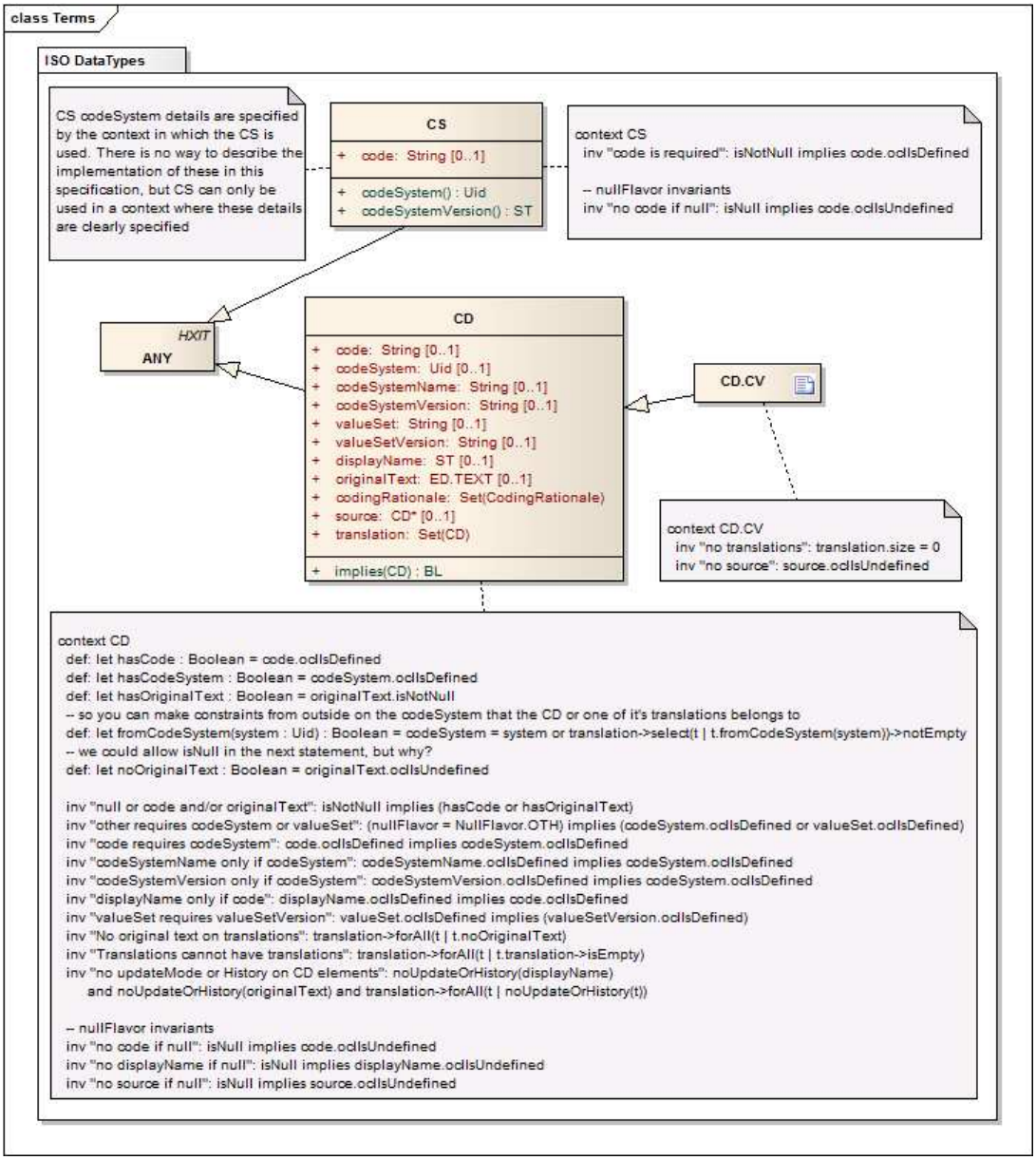


Figure 4. Coded data types

7.5.1 CD (Concept Descriptor)

7.5.1.1 Description

Specializes ANY

A *CD* may contain a simple code - that is, a reference to a concept defined directly by the referenced code system, or it may contain an expression in some syntax defined by the

© ISO 2007 – All rights reserved

referenced code system that can be meaningfully evaluated. e.g., the concept of a "left foot" as a postcoordinated term built from the primary code "FOOT" and the qualifier "LEFT".

A CD may also contain an original text or phrase that served as the basis of the coding. This is preserved to allow for validation of the representation of the concept in various fashions. If the concept (the intent of the original text) cannot be properly represented, then the nullFlavor of the CD should be set to "OTH" (other) and either a reference to a code system or a value set (which may refer to multiple code systems) SHALL be provided.

A CD can contain one or more translations into multiple coding systems. The translations are all representations of the same concept in various code systems. A code system or a value set (which may refer to multiple code systems) SHALL be provided to represent the translation chain - which CD was translated from which - if desired. Each CD may also carry a rationale to indicate why it is represented.

A CD with no nullFlavor attribute SHALL have a code attribute or nonNull originalText attribute. A CD that has a code, codeSystem or originalText attribute but does not meet external constraints on the codeSystem SHALL have a nullFlavor attribute with a value of "OTH".

Attributes with type CD are generally bound by externally specified constraints which constrain the coded concepts to which a CD may refer. These constraints may be qualified as "extensible“ (CWE) or "not extensible“ (CNE). If the constraint is not extensible (CNE), then a the CD that does not have a nullFlavor SHALL contain a code that conforms to the constraint. If the constraint is extensible (CWE) then a CD that does not have a nullFlavor SHALL contain either a code that exists in the domain with which the attribute is associated, a code from a locally defined code system, or just some originalText that describes the concept. If the code is taken from a locally defined code system, then the codeSystem property SHALL specify the local code system.

For both CNE and CWE constraint types, the translations may contain nonNull codes from any source unless otherwise specified by the constraining model.

7.5.1.2 CD and SC

The CD and SC types have very similar structures. CD has a code:codeSystem pair with translations, and an originalText which has type ED.Text – plain text which may be a reference. SC has a string and a code : CD to allow the string to be coded. In SC, the code does not have an originalText – it is fixed to the value attribute of the SC. Therefore both types have a code:codeSystem pair with translations and originalText.

While the types therefore share the same capability and the same core information structure, they differ in purpose. CD exists to provide a reference to a concept in defined in some code system. When a CD is mandatory in the context of use, a code must be provided, and an original text is optional, except in the CWE case discussed above. SC exists to provide string content that may be encoded. When SC is mandatory, the text which becomes the original text must be provided, and the code is always optional.

When it is obvious that one or the other aspect of the coded text is mandatory, then it will be obvious whether to use CD or SC. On the other hand, when neither aspect is mandatory, or both are mandatory, then it is not so obvious which to use. Note that when both are mandatory, either class can be constrained to require both aspects using some formal constraint language such as OCL.

In these cases, it may be ambiguous which of SC and CD is the correct choice. When choosing between them, users should consider the specialization heirarchy, which may dictate a particular choice, and also should consider the constraints that have been made on the type in the context of use; this may also dictate a particular choice.

© ISO 2007 – All rights reserved

In some circumstances it is still ambiguous which type to use. In cases of doubt, implementors should choose CD by default.

7.5.1.3 ISO 11404 Syntax:

```
    type CD = class (
      validTimeLow : characterstring,
      validTimeHigh : characterstring,
      controlActRoot : characterstring,
      controlActExtension : characterstring,
      nullFlavor : NullFlavor,
      updateMode : UpdateMode,
      flavorId : characterstring,
      code : characterstring,
      codeSystem : characterstring,
      codeSystemName : characterstring,
      codeSystemVersion : characterstring,
      valueSet : characterstring,
      valueSetVersion : characterstring,
      displayName : ST,
      originalText : ED.TEXT,
      codingRationale : Set(CodingRationale),
      translation : Set(CD),
      source : CD
   )
```

7.5.1.4 Attributes

7.5.1.4.1 code : String: The plain code symbol defined by the code system, or an expression in a syntax defined by the code system which describes the concept.

If a code is provided, it SHALL be an exact match to a plain code symbol or expression defined by the code system. If the code system defines a code or expression that includes whitespace, the code SHALL include the whitespace. An expression can only be used where the codeSystem either defines an expression syntax, or there is a generally accepted syntax for the codeSystem. A code system may be defined that only defines an expression syntax with bindings to other code Systems for the elements of the expression.

It is at the discretion of the interpreting system whether to check for an expression instead of a simple code and evaluate the expression instead of treating the expression as a code. In some cases, it may be unclear or ambiguous whether the code represents a single symbol or an expression. This usually arises where the code system defines an expression language and then defines pre-coordinated concepts with symbols which match their expression, e.g. UCUM. In other cases, it is safe to treat the expression as a symbol. There is no guarantee that this is always safe: the definitions of the codeSystem should always be consulted to determine how to handle potential expressions.

7.5.1.4.2 codeSystem : Uid: Specifies the code system that defines the code, or if no code was found, the codeSystem in which no code was found.

Code systems SHALL be referred to by a UID, which allows unambiguous reference to standard code systems and other local codesystems. Where either ISO or HL7 have assigned UID to code Systems, then these UIDs SHALL be used. Otherwise implementations SHALL use an appropriate ISO Object Identifier (OID) or UUID to construct a globally unique local coding system identifier.

© ISO 2007 – All rights reserved

A *CD* that has a code attributes SHALL have a codeSystem specifying the system of concepts that defines the code.

An *CD* with a nullFlavor OTH indicates that a concept could not be coded in the coding system or value set specified. Thus, for these coding exceptions, the code system or value set that did not contain the appropriate concept SHALL be provided in codeSystem or valueSet.

7.5.1.4.3 codeSystemName : String: The common name of the coding system.

The code system name has no computational value. *codeSystemName* can never modify the meaning of codeSystem and cannot exist without codeSystem.

Note: The purpose of a code system name is to assist an unaided human interpreter of a code value to interpret codeSystem.

Information Processing Entities claiming direct or indirect conformance SHALL NOT functionally rely on *codeSystemName*. In addition, they MAY choose not to implement *codeSystemName* but SHALL NOT reject instances because *codeSystemName* is present.

7.5.1.4.4 codeSystemVersion : String: If applicable, a version descriptor defined specifically for the given code system.

Note: Different versions of one code system must be compatible. By definition a code symbol SHALL have the same meaning throughout all versions of a code system. Between versions, codes may be retired but not withdrawn or reused. Where the definition of the meaning of a code symbol changes, it must still be compatible (equal) between different code system versions.

Whenever a code system changes in an incompatible way, it will constitute a new code system, not simply a different version, regardless of how the vocabulary publisher calls it. For example, the publisher of ICD-9 and ICD-10 calls these code systems, "revision 9" and "revision 10" respectively. However, ICD-10 is a complete redesign of the ICD code, not a backward compatible version. Therefore, for the purpose of this data type specification, ICD-9 and ICD-10 are different code systems, not just different versions. By contrast, when LOINC updates from revision "1.0j" to "1.0k", this would be considered as just another version of LOINC, since LOINC revisions are backwards compatible.

7.5.1.4.5 valueSet : String: Specifies the valueSet in which no code was found.

Value sets shall be referred to by an identifier name which allows unambiguous reference to a value set. Where either ISO or HL7 have assigned an identifying name to a value set, then that name shall be used.
In many cases, a CD is created from a value set - either a code/code system pair is chosen from a valueSet, or one is not chosen and the CD has the exceptional value of NullFlavor.OTH. If no code is chosen, it is generally inappropriate to reference the code system from which the code was chosen as the value set may not match the code system (may include a subset of the codeSystem, or additional terms from other code systems); instead, the value set should be provided. In addition, there are some known use cases where the value set that a user or system was offered when choosing a code affects the interpretation of the code.

Note that if a code is provided, the meaning of the code must come from the definition of the code in the code system. The meaning of the code SHALL NOT depend on the value set. Information Processing Entities claiming direct or indirect conformance SHALL NOT be required to interpret the code in light of the valueSet, and they SHALL NOT reject an instance because of the presence or absence of any or a particular value set.

© ISO 2007 – All rights reserved

7.5.1.4.6 valueSetVersion : String: Specifies the version of the valueSet in which no code was found.

valueSetVersion SHALL be provided when a valueSet is provided, and otherwise SHALL be null. The valueSet version specifies a date, with the resolution of a particular second, using the format defined for TS.DATETIME.FULL. The date is the date/time that the value set being used was published.

7.5.1.4.7 displayName : ST: A name or title for the code, under which the sending system shows the code value to its users.

If populated, the *displayName* SHALL be a valid human readable representation of the concept as defined by the code system at the time of data entry. The *displayName* SHALL conform to any rules defined by the codingSystem; if the codeSystem does not define a human representation for the code or expression, then none can be provided. *displayName* is included both as a courtesy to an unaided human interpreter of a code value and as a documentation of the name used to display the concept to the user. The display name has no functional meaning; it SHALL never exist without a code; and it SHALL never modify the meaning of the code. A display name may not be present if the code is an expression for which no display name has been assigned or can be derived. Information Processing Entities claiming direct or indirect conformance MAY choose not to implement *displayName* but SHALL NOT reject instances because *displayName* is present.

Note: Display names may not alter the meaning of the code value. Therefore, display names SHOULD NOT be presented to the user on a receiving application system without ascertaining that the display name adequately represents the concept referred to by the code value. Communication SHALL NOT simply rely on the display name. The display name's main purpose is to support implementation debugging.

7.5.1.4.8 originalText : ED.TEXT: The text as seen and/or selected by the user who entered the data.

Original text can be used in a structured user interface to capture what the user saw as a representation of the code on the data input screen, or in a situation where the user dictates or directly enters text, it is the text entered or uttered by the user.

It is valid to use the CD data type to store only the text that the user entered or uttered. In this situation, original text will exist without a code. In a situation where the code is assigned sometime after the text was entered, originalText is the text or phrase used as the basis for assigning the code.

Notes

- The details of the link in the originalText.reference between different artifacts of medical information (e.g., document and coded result) is outside the scope of this specification and may be further proscribed in specifications that use this specification.

  - The original text SHALL be an excerpt of the relevant information in the original sources, rather than a pointer or exact reproduction. Thus the original text SHALL be represented in plain text form. In specific circumstances, when clearly descirbed the context of use, the originalText may be a reference to some other text artefact for which the resolution scope is clearly described.

  - Values of type CD MAY have a original text despite not having a code. Any CD value with no code signifies a coding exception. In this case, originalText is a name or description of the concept that was not coded. Such CD values MAY also contain translations.

© ISO 2007 – All rights reserved

- Translations directly encode the concept described in originalText. The originalText represents the originalText of the concept itself. Translations SHALL NOT have an originalText of their own.

7.5.1.4.9 translation : Set(CD): A set of other CDs that each represent a translation of this CD into equivalent codes within the same code system or into corresponding concepts from other code systems.

NOTE: The translations are quasi-synonyms of one real-world concept. Every translation in the set is supposed to express the same meaning "in other words." However, exact synonymy rarely exists between two structurally different coding systems. For this reason, not all of the translations will be equally exact.

Translations SHALL NOT contain translations. The root CD has one set of translations which lists all the translations. The root translation is generally the one that best meets the conformance criteria for the CD. No implication about lineage of the translations can be drawn from the selection of the root code. Instead the properties codingRationale and source is used to trace lineage.

In the absence of a constraining model that makes constraints on the value domain of the CD, any of the translations MAY be the root CD. If the constraining model makes constraints on the value domain of the CD and there is a translation that meets the constraints, that translation SHOULD be the root CD. If the constraining model makes constraints on the value domain of the CD and there is no translation that meets the constraints, then any of the translations MAY be the root, as long as they are assigned a nullFlavor. An alternative is to put none of the translations in the root, and give it a nullFlavor of choice, and put all the translations in the translation property of the root.

7.5.1.4.10 codingRationale : Set(CodingRationale): provides a reason why a particular *CD* has been provided, either as the root concept or as one of the translations.

If populated, the values contained in this attribute SHALL be taken from the HL7 CodingRationale code system. The current values are:

CodingRationale Enumeration. OID: 2.16.840.1.113883.5.1074			
1	O	original	Originally produced code
1	P	post-coded	post-coded from free text source
1	R	required	Required by the specification describing the use of the coded concept. The exact form of the requirement is not specified here; it may be required by the specification directly, or it may arise as an indirect result of other conformance tools. More than one different requirement may exist simultaineously, so more than one code in a CD complex may be required

11404 Syntax:

```
type CodingRationale = enumeration (O, P, R)
```

The values O and P are mutually exclusive – only one of these two values SHALL be part of codingRationale at any given time. A code is deemed to be post-coded if the user does not review the choice of the code when they first enter the data. codingRationale is not expected to act as a quality review marker on the quality of the coding or the translation processes.

A code is required when it is present in the instance to meet some constraints imposed on the instance by the context of use. Information Processing Entities SHALL not be required to mark a particular translation as required even though it is required by the context of use, but MAY do so. Information Processing Entities SHALL not reject instances because of the presence or absence of the codingRationale flag.

© ISO 2007 – All rights reserved

7.5.1.4.11 source : CD: A reference to the CD that was the source of this translation, if this CD was created by translating it from another CD. This property is a reference, and if used, must be a reference to a CD within the same cluster of CDs that represent the same concept. Specifically, the source SHALL be provided within the scope of this CD's root CD and translations.

A CD consists of a single root code and a set of translations, which do not have translations. Using the codingRationale property, a sender can indicate which is the original code. There are some circumstances in which it is useful to know which CD was translation from which CD. The source allows for the translation sequence from one translation to another to be represented. Each element of the translation set was translated from the original CD. Each translation may, however, also have further translations. Thus, when a code is translated multiple times the information about which code served as the input to which translation will be preserved.

7.5.1.5 Equality

The equality of two CD values is determined solely based upon code and codeSystem. The codeSystemVersion, originalText, codingRationale, source, valueset information and translations are not included in the equality test. Null values are not equal even if they have the same NULL-flavor or the same original text.

The equality is based on the literal value of the code and codeSystem. Information Processing Entiies SHALL not consult the semantic meaning of the code/codeSystem pair to determine whether the same concept is intended.

Note 1: this means that for SNOMED, for example, two isomorphic forms of the same expression will not be equal. Implementations should choose the literal representation forms carefully.

Note 2: CD values may also be equal with values of the CS datatype.

Note 3: CD values may also be equal with values of the CO datatype. For details, consult section 7.8.5.4.

7.5.1.6 Invariants

- if the value is not null then code must have a value
- if code has a value then codeSystem must have a value
- valueSet requires a valueSetVersion
- codeSystemName can only have a value if codeSystem has a value
- codeSystemVersion can only have a value if codeSystem has a value
- displayName can only have a value if code has a value
- translations cannot have original text
- translations cannot have translations

OCL for invariants:

© ISO 2007 – All rights reserved

```
def: let hasCode : Boolean = code.oclIsDefined
def: let hasCodeSystem : Boolean =
    codeSystem.oclIsDefined
def: let hasOriginalText : Boolean =
    originalText.isNotNull
def: let fromCodeSystem(system : Uid) : Boolean =

    codeSystem = system or translation->select(t |
    t.fromCodeSystem(system))->notEmpty
```

Note: fromCodeSystem is defined so that you can make constraints from outside on the codeSystem of the CD or one of it's translations. i.e. inv: code.fromCodeSystem("2.16.840.1.113883.6.42")

```
def: let noOriginalText : Boolean
     originalText.oclIsUndefined

 inv "null or code and/or originalText": isNotNull
      implies (hasCode or hasOriginalText)
 inv "other requires codeSystem or valueSet":
     (nullFlavor = NullFlavor.OTH) implies
     (codeSystem.oclIsDefined or valueSet.oclIsDefined)
inv "code requires codeSystem": code.oclIsDefined
    implies codeSystem.oclIsDefined
inv "codeSystemName only if codeSystem":
     codeSystemName.oclIsDefined implies
     codeSystem.oclIsDefined
inv "codeSystemVersion only if codeSystem":
     codeSystemVersion.oclIsDefined implies
     codeSystem.oclIsDefined
inv "displayName only if code":displayName.oclIsDefined
   implies code.oclIsDefined
inv "valueSet requires valueSetVersion":
  valueSet.oclIsDefined implies
     (valueSetVersion.oclIsDefined)
inv "No original text on translations": translation->
    forAll(t | t.noOriginalText)
inv "Translations cannot have translations":
    translation->forAll(t | t.translation->isEmpty)
inv "no updateMode or History on CD elements":
   noUpdateOrHistory(displayName)and
     noUpdateOrHistory(originalText) and
    translation->forAll(t | noUpdateOrHistory(t))
```

Null flavor invariants

```
inv "no code if null": isNull implies
     code.oclIsUndefined
inv "no displayName if null": isNull implies
   displayName.oclIsUndefined
inv "no source if null": isNull implies
    source.oclIsUndefined
```


© ISO 2007 – All rights reserved

7.5.1.7 Operations

7.5.1.7.1 implies(other : CD):BL: True if this code/codeSystem is a specialization of the other code/codeSystem or has the same meaning.

Note: in SNOMED, for example, two isomorphic forms of the same expression will imply each other.

Note: A terminology service may be used to make this determination.

7.5.1.8 Examples

7.5.1.8.1 ICD Examples

A simple example for code is the ICD-9 code for headache, which is "784.0".

```
<example code="784.0" codeSystem="2.16.840.1.113883.6.42"
       codeSystemName="ICD-9">
  <displayName value="Headache"/>
  <originalText value="general headache"/>
</example>
```

A possible ICD-10 equivalent is "G44.1" (the ICD-10 classifications are slightly different).

```
<example code="G44.1" codeSystem="2.16.840.1.113883.6.3"
       codeSystemName="ICD-10">
  <displayName value="Headache"/>
  <originalText value="general headache"/>
</example>
```

7.5.1.8.2 Coding Failure Examples

A common situation with CD is when the actual concept cannot be properly represented in a particular coding system. Usually this circumstance arises where the concept is expected to be represented in a particular coding system. For the purposes of these examples, we assume that all these examples are for an observation value of type CD that is bound to the full Snomed-CT valueset (Example OID for the value set = 2.16.840.1.113883.19.11.1 as published 11-June 2007, Real OID for the SNOMED-CT code system = 2.16.840.1.113883.6.96). *Important Note: The OID root 2.16.840.1.113883.19 is an HL7 OID used for example-only OIDs and OIDs in this space are never valid in real instances. The OIDs used in these examples that in the OID space 2.16.840.1.113883.6, 2.16.840.1.113883.5 and 2.16.840.1.113883.11 are the correct OIDs for use in production instances.*

The simplest case is where the CD is not represented in the instance at all, or simply represented as no information.

```
<value nullFlavor="NI"/>
```

However this isn't a very useful representation - frequently the source system knows more information, and it is still useful to convey that information to the destination system, while still labelling the coding as incomplete.

```
<value nullFlavor="OTH" codeSystem="2.16.840.1.113883.6.96"/>
```

or it may be encoded as

```
<value nullFlavor="OTH" valueSet="2.16.840.1.113883.19.11.1"
```

© ISO 2007 – All rights reserved

```
valueSetVersion="20070711""/>
```

Both examples say that the concept cannot be coded in SNOMED. Even more useful is to convey some specific information about the concept, even though it cannot be represented in SNOMED:

```
<value nullFlavor="OTH" codeSystem="2.16.840.1.113883.6.96">
 <originalText value="Burnt ear with iron. Burnt other ear calling
for ambulance"/>
</value>
```

It's also possible that the content was first encoded in some other code system than SNOMED, and the source system was unable to encode the value in SNOMED. In this case, there is two forms of representation. The first is when the binding to SNOMED is labelled as CWE: local extensions are allowed:

```
<value code="burn" codeSystem="2.16.840.1.113883.19.5.2">
 <originalText value="Burnt ear with iron. Burnt other ear calling
for ambulance"/>
</value>
```

In this case, because the binding is CWE, local extensions are allowed, and the source system can simply use it's own codeSystem (here identified by the OID "2.16.840.1.113883.19.5.2", which is an example OID) to extend the other code system. In fact, the source system can also use a code from another well known code system, such as ICD-9. If ICD-9 had a code "A10.1" which stood for this same concept, then this would be valid:

```
<value code="A10.1" codeSystem="2.16.840.1.113883.6.42">
  <originalText value="Burnt ear with iron. Burnt other ear
   calling for ambulance"/>
</value>
```

If, however, the binding to the SNOMED-CT valueset is labelled CNE, then the code must come from SNOMED. The same information as the case above must be conveyed differently:

```
<value nullFlavor="OTH" codeSystem="2.16.840.1.113883.6.96">
  <originalText value="Burnt ear with iron. Burnt other ear
   calling for ambulance"/>
  <translation code="burn" codeSystem="2.16.840.1.113883.19.5.2">
</value>
```

Now the code is clearly marked as OTH: the code cannot be represented in SNOMED-CT, but a translation from another system is provided. Though it's pretty redundant in this case, the source system could indicate which translation comes from which using the source property:

```
<value nullFlavor="OTH" codeSystem="2.16.840.1.113883.6.96"
    source="s1">
  <originalText value="Burnt ear with iron. Burnt other ear
    calling for ambulance"/>
  <translation id="s1" code="burn"
    codeSystem="2.16.840.1.113883.19.5.2">
</value>
```

All these examples have assumed that the attribute is bound to the fictitious value set 2.16.840.1.113883.19.11.1 which is all of SNOMED-CT. If the value set was extended to include the LOINC codes as well, it would no longer be appropriate to encode a failure to code like this:

```
<value nullFlavor="OTH" codeSystem="2.16.840.1.113883.6.96"/>
```

since it's not true that the concept could not be coded from SNOMED-CT - it could not be coded in either SNOMED-CT or LOINC. For this reason, it is appropriate to encode the failure to code in the valueSet form:

© ISO 2007 – All rights reserved

```
<value nullFlavor="OTH" valueSet="2.16.840.1.113883.19.11.1"
valueSetVersion="20070711"/>
```

7.5.1.8.3 Expression examples

Expressions generally arise with complex medical terminologies such as SNOMED. For example, SNOMED CT defines a concept "cellulitis (disorder)" (128045006) an attribute "finding site" (363698007) and another concept "foot structure (body structure)" (56459004). SNOMED CT allows these codes to be combined in a code phrase:

```
128045006:{363698007=56459004}
```

The full CD form for this is:

```
<value code="128045006:{363698007=56459004}"
    codeSystem="2.16.840.1.113883.6.42" codeSystemName="Snomed-CT">
  <originalText value="Cellulitis of the foot"/>
</value>
```

The SNOMED compositional expression language is currently undergoing comment, and may be found on the IHTSDO website (http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/Technical_Docs/abstract_models_and_representational_forms.pdf). The next two examples are based on SNOMED CT Core Edition 2007-01-31.

This first example is the SNOMED code for "fracture of left tibia". It shows issues associated with grouping and nesting.

```
31978002|fracture of tibia|: 272741003|laterality|=7771000|left|
```

Strictly speaking (in normal form) a "fracture of left tibia" is not a "left fracture" of a "tibia bone" but is a "fracture" of the "left" "tibia bone" (that is, the qualification of "left" applies to the bone not to the fracture). Also note in this example that the fracture and bone are grouped - this may look irrelevant but is potentially significant for combined fractures where different morphology may apply to different bones. An alternative rendering for this same concept is:

```
64572001|disease|:{116676008|associated morphology|=72704001|fracture|,
  363698007|finding site|=(12611008|bone structure of
    tibia|:272741003|laterality|=7771000|left|)}
```

The second example shows a more complicated grouping and nesting structure. The SNOMED CT expression for "past history of fracture of left tibia" includes nesting even in it's simplest form because the laterality does not apply to the past history but rather to the disorder.

```
417662000|past history of clinical finding|:246090004|associated finding|=
     (31978002|fracture of tibia|: 272741003|laterality|=7771000|left|)
```

The alternative rendering is even more nested:

```
243796009|situation with explicit context|:246090004|associated finding|=
   (64572001|disease|:{116676008|associated morphology|=72704001|fracture|,
   363698007|finding site|=(12611008|bone structure of tibia|:
   272741003|laterality|=7771000|left|)}),408729009|finding context|=
   410515003|known present|,408731000|temporal context|=410513005|past|,
   408732007|subject relationship context|=410604004|subject of record|
```

These are provided as examples of SNOMED expression syntaxes. A full discussion the merits of the different forms, their relationship and how to work with them can be found in the SNOMED compositional expression language definition referred to above.

It is important to note that the expression syntax and semantic rules are specified by the code system. For instance, in SNOMED CT, there are a defined set of qualifying attributes, and

© ISO 2007 – All rights reserved

only Findings and Disorders can be qualified with the "finding site" attribute. CD does not provide for normalization of compositional expressions, therefore it is possible to create ambiguous expressions. Users should understand that they must provide the additional constraints necessary to assure unambiguous data representation, if they are planning to create compositional expressions using CD. Otherwise, they risk the inability to retrieve a complete set of all records corresponding to any given query.

ICD-10 allows dual coding. Refer to Section 3.1.3 of the ICD-10 Instruction Manual (2nd Edition, found at http://www.who.int/entity/classifications/icd/ICD-10_2nd_ed_volume2.pdf). While ICD-10 clearly establishes the semantic basis for the dual coding, it does not define an actual literal expression form suitable for use with CD. In such cases, HL7 defines a suitable literal expression form and assigns an OID to that. The OID for this ICD-10 expression is 2.16.840.1.113883.6.260. The code system specifies that the two ICD-10 codes are separated by a space.

```
<value code="J21.8 B95.6" codeSystem="2.16.840.1.113883.6.260"
      codeSystemName="ICD-10 Dual Code Expression">
  <originalText value="Staph aureus bronchiolitis"/>
</value>
```

The ICD-10 code J21.8 is "Acute bronchiolitis due to other specified organisms" and the code B95.6 is "Staphylococcus aureus as the cause of diseases classified to other chapters".

7.5.2 CD.CV (Coded Value)

7.5.2.1 Description

Constrains CD

Coded data, specifying only a code, code system, and optionally display name and original text.

Used only as the type of properties of other datatypes.

CV is used when any reasonable use case will require only a single code value to be sent. Thus, it should not be used in circumstances where multiple alternative codes for a given value are desired or there may be a requirement to migrate to a new coding system.

7.5.2.2 Invariants

- No translations are allowed
- No source is allowed

OCL for invariants:

```
inv "no translations": translation.size = 0
inv "no source": source.oclIsUndefined
```


© ISO 2007 – All rights reserved

7.5.3 CS (Coded Simple Value)

7.5.3.1 Description

Specializes ANY

Coded data in its simplest form, where only the code is not predetermined.

The code system and code system version are implied and fixed by the context in which the CS value occurs. Due to it's highly restricted functionality, CS SHALL only be used for simple structural attributes with highly controlled and stable terminologies.

7.5.3.2 11404 Syntax:

```
type CS = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  code : characterstring
)
```

7.5.3.3 Attributes

7.5.3.3.1 code : String: The plain code symbol defined by the code system. if the code value is empty or null, then there is no code in the code system that represents the concept.

Code SHALL only contain characters that are either a letter, a digit, or one of '.', '-', '_' or ':'. Code systems that are used with *CS* SHALL NOT define code symbols or expression syntaxes that contain whitespace or any other characters not in this list.

7.5.3.4 Equality

The equality of two CS values is determined solely based upon the explicit code and the implicit codeSystem. The codeSystemVersion.is not included in the equality test. Null values are not equal even if they have the same NULL-flavor

The equality is based on the literal value of the code and codeSystem. Information Processing Entiies SHALL not consult the semantic meaning of the code/codeSystem pair to determine whether the same concept is intended.

Note: CS values may be equal to CD values if both specify the same code and codeSystem.

7.5.3.5 Invariants

- There must be a code if not null

OCL for invariants:

```
inv "code is required": isNotNull implies
```

© ISO 2007 – All rights reserved

```
        code.oclIsDefined

  nullFlavor invariants
      inv "no code if null": isNull implies
        code.oclIsUndefined
```

7.5.3.6 Operations

7.5.3.6.1 codeSystem() : Uid: Although a CS does not carry a codeSystem as an attribute, there must always be an applicable codeSystem specified by the context in which CS is used. This operation returns the codeSystem that is specified by the context.

This operation must always return a valid codeSystem, whether or not the CS is null, as it is taken from the context of use.

7.5.3.6.2 codeSystemVersion() : String: Although a CS does not carry a codeSystemVersion as an attribute, there may be an applicable version specified by the context in which CS is used.

This operation returns the codeSystemVersion if one is specified by the context. This operation must always return a valid codeSystem, whether or not the CD is null, as it is taken from the context of use.

7.5.3.7 Examples

```
<code xsi:type="CS" code="NS"/>
```

A simple code NS in the designated code system.

© ISO 2007 – All rights reserved

7.6 Identification and Location Datatypes

These data types provide support for identifying objects, records, and things, and specifically for URLs, URIs and telecommunication addresses.

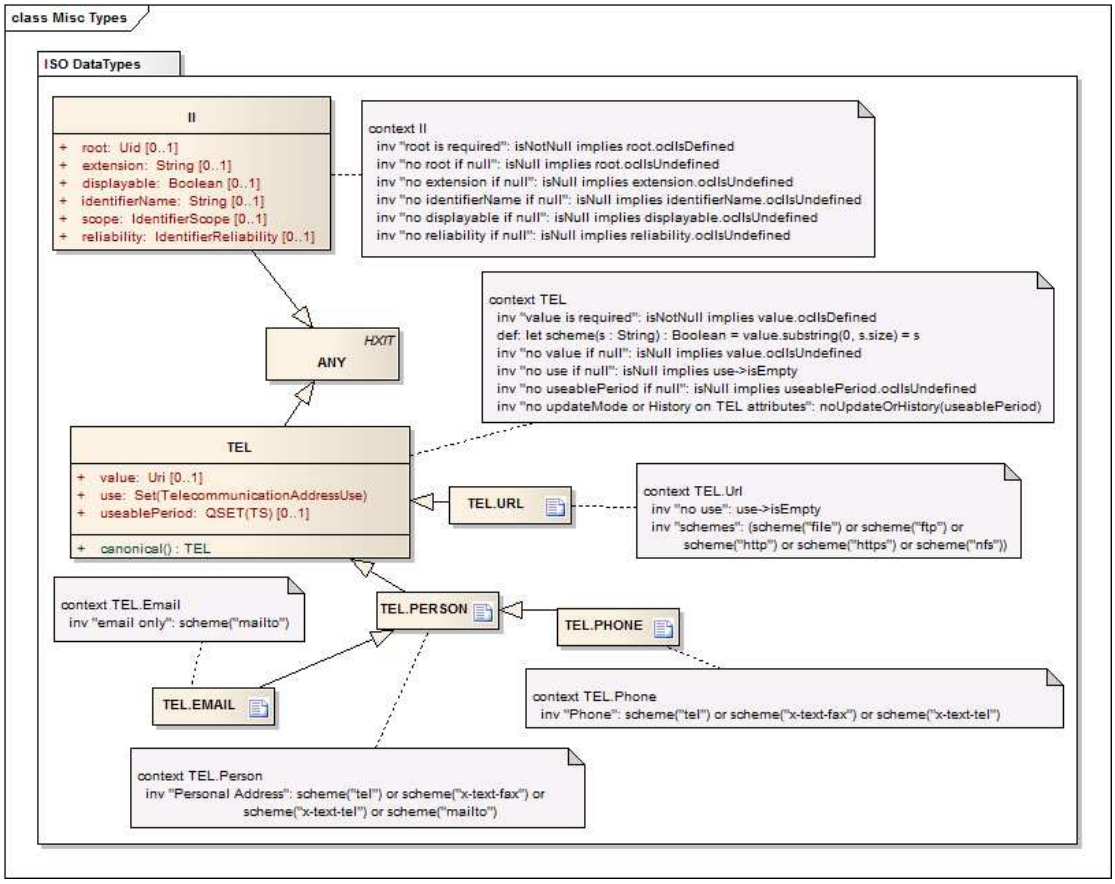


Figure 5. Identification and location data types

7.6.1 TEL (Telecommunication Address)

7.6.1.1 Description

Specializes ANY

A locatable resource that is identified by a URI, such as a web page, a telephone number (voice, fax or some other resource mediated by telecommunication equipment), an e-mail address, or any other locatable resource that can be specified by a URL.

The address is specified as a Universal Resource Locator (URL) qualified by time specification and use codes that help in deciding which address to use for a given time and purpose.

Note 1: The value attribute is constrained to be a uniform resource locator specified according to IETF RFCs 1738 and 2806 when used in this data type.

© ISO 2007 – All rights reserved

Note 2: The intent of this data type is to be a locator, not an identifier; this data type is used to refer to a locatable resource using a URL, and knowing the URI allows one to locate the object. However some use cases have arisen where a URI is used to refer to a locatable resource. Though this data type allows for URIs to be used, the resource identified SHOULD always be locatable. A common use of URI's is to refer to SOAP attachments.

7.6.1.2 ISO 11404 Syntax

```
type TEL = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  value : characterstring,
  use : Set(TelecommunicationAddressUse),
  useablePeriod : QSET(TS)
)
```

7.6.1.3 Attributes

7.6.1.3.1 value : Uri: A uniform resource identifier specified according to IETF RFC 2396.

The URI specifies the protocol and the contact point defined by that protocol for the resource.

Examples: Notable uses of the telecommunication address datatype are for telephone and telefax numbers, e-mail addresses, Hypertext references, FTP references, etc.

7.6.1.3.2 use : Set(TelecommunicationAddressUse): One or more codes advising system or user which telecommunication address in a set of like addresses to select for a given telecommunication need.

Note: The telecommunication use code is not a complete classification for equipment types or locations. Its main purpose is to suggest or discourage the use of a particular telecommunication address. There are no easily defined rules that govern the selection of a telecommunication address.

If populated, the values contained in this attribute SHALL be taken from the HL7 TelecommunicationAddressUse code system. The current values are:

TelecommunicationAddressUse Enumeration. OID: 2.16.840.1.113883.5.1011			
1	H	home address	A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available
2	HP	primary home	The primary home, to reach a person after business hours
2	HV	vacation home	A vacation home, to reach a person while on vacation
1	WP	work place	An office address. First choice for business related contacts during business hours
2	DIR	direct	Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries.

© ISO 2007 – All rights reserved

			For phones, often referred to as a ‘private line’
2	PUB	public	Indicates a work place address or telecommunication address that is a ‘standard’ address which may reach a reception service, mail-room, or other intermediary prior to the target entity
1	BAD	bad address	A flag indicating that the address is bad, in fact, useless
1	TMP	temporary address	A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information
1	AS	answering service	An automated answering machine used for less urgent cases and if the main purpose of contact is to leave a message or access an automated announcement
1	EC	emergency contact	A contact specifically designated to be used for emergencies. This is the first choice in emergencies, independent of any other use codes
1	MC	mobile contact	A telecommunication device that moves and stays with its owner. May have characteristics of all other use codes, suitable for urgent matters, not the first choice for routine business
1	PG	pager	A paging device suitable to solicit a callback or to leave a very short message

ISO 11404 Syntax for telecommunicationAddressUse attribute

```
 type TelecommunicationAddressUse = enumeration (H, HP, HV,
WP, DIR, PUB, BAD, TMP, AS, EC, MC, PG)
```

7.6.1.3.3 useablePeriod: QSET(TS): Specifies the periods of time during which the telecommunication address can be used.

Note: For a telephone number, this can indicate the time of day in which the party can be reached on that telephone. For a web address, it may specify a time range in which the web content is promised to be available under the given address.

7.6.1.4 Equality

Two nonNull TEL values are equal if they have the same value attribute. Separator characters SHALL not be ignored when comparing *URL*s with tel: scheme. The use and useablePeriod attributes are excluded from the equality test.

7.6.1.5 Invariants

- value must be provided

OCL for Invariants:

```
inv "value is required": isNotNull implies
   value.oclIsDefined
def: let scheme(s : String) : Boolean
      = value.substring(0, s.size) = s
inv "no updateMode or History on TEL attributes":
```

© ISO 2007 – All rights reserved

```
            noUpdateOrHistory(useablePeriod)

nullflavor invariants:
        inv "no use if null": isNull implies use->size = 0
        inv "no useablePeriod if null": isNull implies
         (useablePeriod.oclIsUndefined or useablePeriod.isNull)
        inv "no value if null": isNull implies
           value.oclIsUndefined
```

7.6.1.6 Operations

7.6.1.6.1 canonical : TEL: The TEL with any seperator or other non-significant characters stripped out of the address.

Note: The tel: syntax allows for characters such as (and) which are syntactical separator characters but do not change the actual telephone number. canonical strips characters like these out of the address portion. The actual characters stripped out depends on the scheme.

7.6.1.7 Extensions to URL/URI syntax

This specification defines the following extensions to the **URL** scheme:

- x-text-tel: - indicates that the destination device is a text telephone. The syntax of the address portion of the URL is the same as TEL
- x-text-fax: - indicates that the destination device is a fax machine. The syntax of the address portion of the URL is the same as TEL. This protocol replaces the deprecated W3C protocol fax:

This specification defines the following extensions to the **URN** scheme:

- hl7ii – a reference to an II value defined in this specification. The full syntax of the URN is urn:hl7ii:{root}[:{extension}] where {root} and {extension} (if present) are the values from the II that is being referenced. Full details of this protocol are defined in the HL7 Abstract Datatypes Specification.

7.6.1.8 Examples

7.6.1.8.1 Web address

```
<example xsi:type="TEL" value="http://www.temp.org/example/234232"/>
```

A reference to the web page available from `http://www.temp.org/example/234232` .

7.6.1.8.2 Combined home and work phone

```
<example xsi:type="TEL" value="tel:+15556755745" use="H WP"/>
```

A home (H) phone number for a person who works at home (WP).

© ISO 2007 – All rights reserved

7.6.1.8.3 Unknown home phone number

```
<example xsi:type="TEL" nullFlavor="UNK" value="tel:" use="H"/>
```

An unknown home (H) phone number.

7.6.1.8.4 Work phone with extension

```
<example xsi:type="TEL" value="tel:+1(555)6755745;postd=545"
use="WP"/>
```

A work phone with an extension specified. Note that extensions are not the only use for the post-dial sequence. Consult RFC 2806 [http://www.ietf.org/rfc/rfc2806.txt] for further details. The canonical form of this example is:

```
<tel value="tel:+15556755745;postd=545" use="WP"/>
```

7.6.2 TEL.URL

7.6.2.1 Description

Constrains TEL

TEL.URL constrains TEL so that it must point to a locatable resource that returns binary content.

7.6.2.2 Invariants

- No use codes
- The URL scheme must be file, nfs, ftp, cid (for SOAP attachments), http, or https.

OCL for Invariants:

```
inv "no use": use->isEmpty
inv "schemes": (scheme("file") or scheme("ftp") or
   scheme("cid") or scheme("http") or scheme("https")
    or scheme("nfs"))
```

7.6.3 TEL.PERSON

7.6.3.1 Description

Constrains TEL

TEL.PERSON constrains TEL so that it must refer to a method of communication with a person.

7.6.3.2 Invariants

- The URL scheme must be tel, x-text-fax, x-text-tel or mailto

© ISO 2007 – All rights reserved

OCL for Invariants:

```
inv "Personal Address": scheme("tel") or
     scheme("x-text-fax")or scheme("x-text-tel") or
     scheme("mailto")
```

7.6.4 TEL.PHONE

7.6.4.1 Description

Constrains TEL.PERSON

TEL.PHONE constrains TEL.PERSON so it must refer to some telephone based communication system with a person.

7.6.4.2 Invariants

- The URL scheme must be tel, x-text-fax, or x-text-tel

OCL for Invariants:

```
inv "Phone": scheme("tel") or scheme("x-text-fax") or
     scheme("x-text-tel")
```

7.6.5 TEL.EMAIL

7.6.5.1 Description

Constrains TEL.PERSON

TEL.EMAIL constriains the TEL.PERSON type to be an email address.

7.6.5.2 Invariants

- The URL scheme must be mailto

OCL for Invariants:

```
inv "email only": scheme("mailto")
```

7.6.6 II (Instance Identifier)

7.6.6.1 Description

Specializes ANY

An identifier that uniquely identifies a thing or object.

Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance identifiers are defined based on ISO object identifiers.

© ISO 2007 – All rights reserved

An identifier allows someone to select one record, object or thing from a set of candidates. Usually an identifier alone without any context is not usable. Identifiers are distinguished from concept descriptors as concept descriptors never identify an individual thing, although there may sometimes be an individual record or object that represents the concept.

Information Processing Entities claiming direct or indirect conformance SHALL never assume that receiving applications can infer the identity of issuing authority or the type of the identifier from the identifier or components thereof.

7.6.6.2 ISO 11404 Syntax:

```
type II = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  root : characterstring,
  extension : characterstring,
  identifierName : characterstring,
  displayable : boolean,
  scope: IdentifierScope,
  reliability : IdentifierReliability
)
```

7.6.6.3 Attributes

7.6.6.3.1 root : Uid: A unique identifier that guarantees the global uniqueness of the instance identifier.

In the presence of a non-null extension, the root is the unique identifier for the "namespace" of the identifier in the extension. Note that this does NOT necessarily correlate with the organization that manages the issuing of the identifiers. A given organization may manage multiple identifier namespaces, and control over a given namespace may transfer from organization to organization over time while the root remains the same.

This field can be either a DCE UUID, an Object Identifier (OID), or a special identifier taken from lists that may be published by ISO or HL7.

Notes:

1. Comparison of root values is always case sensitive. UUID's SHALL be represented in upper case, so UUID case should always be preserved.

2. The root should not be used to carry semantic meaning – all it does is ensure global computational uniqueness.

© ISO 2007 – All rights reserved

7.6.6.3.2 extension: String: A character string as a unique identifier within the scope of the identifier root.

The root and extension scheme means that the concatenation of root and extension SHALL be a globally unique identifier for the item that this II value identifies.

Some identifier schemes define certain style options to their code values. For example, the U.S. Social Security Number (SSN) is normally written with dashes that group the digits into a pattern "123-12-1234". However, the dashes are not meaningful and a SSN can also be represented as "123121234" without the dashes. In the case where identifier schemes provide for multiple representations, HL7 or ISO may make a ruling about which is the preferred form and document that ruling where that respective external identifier scheme is recognized.

7.6.6.3.3 identifierName : String: This is a human-readable name for the namespace represented in the root.

Note: It is a descriptive name for the actual namespace. e.g. "California, U.S. Driver's License Number, 1970-".

IdentifierName does NOT refer to the organization which issued the identifier (e.g. California Dept. of Motor Vehicles). It is intended for use as a human readable label when an identifier must be displayed to a human user where an OID would not be meaningful.

Notes:

1. In general, it should only be used when an extension is present, allowing for a display such as "California, U.S. Driver's License Number, 1970-: 123456789". There are absolutely no guidelines for the contents of this text other than it should be completely descriptive of the namespace. E.g. "Driver's License" or even "California Driver's License" would not be ideal. However, formatting, capitalization, whitepace, language, etc. are completely up to the sender.

2. The Identifier Name has no computational value. IdentifierName can never modify the meaning of the root attribute. The purpose of the Identifer Name is to assist an unaided human interpreter of an II value to interpret the authority. Applications SHALL NOTattempt to perform any decision-making, matching, filtering or other processing based on this presence or value of this property. It is for display and development assistance only. All decision logic MUST be based solely on the root and extension properties. Information Processing Entities claiming direct or indirect conformance MAY choose not to implement *identifierName* but SHALL NOT reject instances because *identifierName* is present.

7.6.6.3.4 displayable : Boolean: Specifies if the identifier is intended for human display and data entry (displayable = true) as opposed to pure machine interoperation (displayable = false).

Information Processing Entities claiming direct or indirect conformance MAY choose not to implement *displayable* but SHALL NOT reject instances because *displayable* is present.

7.6.6.3.5 scope: IdentifierScope : Specifies the scope in which the identifier applies to the object with which it is associated.

If populated, the value of this attribute SHALL be taken from the HL7 IdentifierScope code system. The current values are:

IdentifierScope Enumeration. OID: [not yet assigned]

© ISO 2007 – All rights reserved

1	BUSN	business identifier	An identifier whose scope is defined by business practices associated with the object. In contrast to the other scope identifiers, the scope of the use of the id is not necessarily restricted to a single object, but may be reused for other objects closely associated with the object due to business practice.
1	OBJ	object identifier	The identifier associated with a particular object. It remains consistent as the object undergoes state transitions
1	VER	version identifier	An identifier that references a particular object as it existed at a given point in time. The identifier SHALL change with each state transition on the object. I.e. The version identifier of an object prior to a 'suspend' state transition is distinct from the identifier of the object after the state transition. Each version identifier can be tied to exactly one ControlAct event which brought that version into being (though the control act may never be instantiated). NOTE: Applications that do not support versioning of objects must ignore and not persist these ids to avoid confusion resulting from leaving the same identifier on an object that undergoes changes.
1	VW	view specific identifier	An identifier for a particular snapshot of a version of the object. This identifies a view of the business object at a particular point in time, and as such identifies a set of data items that can be digitally signed and/or attested. This is in contrast to the Version Identifier which identifies the object at a specific time, but not the amount of information being asserted about the object. This identifier would be changed when a transformation of the information is performed (eg to add code translations, to provide a simplified textual rendering, or to provide additional information about the object as it existed at the specific point in time)

© ISO 2007 – All rights reserved

11404 Syntax for identifierScope attribute

```
type IdentifierScope = enumeration (BUSN, OBJ, VER, VW)
```

7.6.6.3.6 reliability: IdentifierReliability : Specifies the reliability with which this identifier is known. This attribute may be used to assist with identifier matching algorithms.

If populated, the value of this attribute SHALL be taken from the HL7 IdentifierReliability code system. The current values are:

IdentifierReliability Enumeration. OID: [not yet assigned]			
1	ISS	issued by system	The identifier was issued by the system responsible for constructing the instance
1	VRF	verified by system	The identifier was not issued by the system responsible for constructing the instance, but the system that captured the id has verified the identifier with the issuing authority, or with another system that has verified the identifier
1	UNV	Unverified by system	The identifier was provided to the system that constructed the instance, but has not been verified. e.g. a Driver's licence entered manually into a system by a user

ISO 11404 syntax for identifierReliability attribute

```
type IdentifierReliability = enumeration (ISS, VRF, USE)
```

7.6.6.4 Equality

Two instance identifiers are equal if and only if their root is equal, and their extensions are both NotApplicable (nullFlavor="NA") or equal. The displayable, identifierName, scope and reliability properties are ignored, though the scope and reliability properties may be used to determine whether the equality is significant in a given context.

7.6.6.5 Invariants

- A root must be present if the II is not null

OCL Invariants:

```
inv "root is required": isNotNull implies root.oclIsDefined
```

nullFlavor invariants

```
inv "no root if null": isNull implies
     root.oclIsUndefined
inv "no extension if null": isNull implies
   extension.oclIsUndefined
inv "no identifierName if null": isNull
     implies identifierName.oclIsUndefined
inv "no displayable if null": isNull implies
   displayable.oclIsUndefined
inv "no reliability if null": isNull implies
   reliability.oclIsUndefined
```


© ISO 2007 – All rights reserved

7.6.6.6 ISO 22220 Comments

ISO 22220 defines four fields for subject of care identifiers: designation, geographic, issuer, and type. Only the first, the designation, matches the scope of the II type. The designation is defined as

> A number or code assigned to a person by an organization, establishment, agency or domain in order to uniquely identify that person as a subject of health care within that health care organization, establishment, agency or domain.

II fulfills this role by providing a unique identifier. When an II is used to identify a subject of care, the context of use should provide support for the scope, issuer and type properties.

Note: the II provides a field called "IdentifierName" but this field does not provide formal support for identifying the issuer of the identifier.

7.6.6.7 Examples

7.6.6.7.1 Drivers License

```
<example xsi:type="II" root="2.16.840.1.113883.12.333"
extension="45634353344" reliability="USE" scope="BUSN"/>
```

The OID 2.16.840.1.113883.12.333 has been issued by HL7 as a generic driver's license authority. The extension contains the actual drivers license number. The reliability is USE - the value has been entered into the system, but the system cannot verify the number. The scope of the driver's license is BUSN, because the driver's license number may be used to identify several different objects associated with the same patient. This example uses the generic oid for a drivers license authority, but it is recommended not to use this in practice if possible because there is many drivers license authorities, and their issuing numbers will clash.

7.6.6.7.2 US SSN

```
<example xsi:type="II" root="2.16.840.1.113883.4.1"
extension="123456789" reliability="USE" scope="BUSN"/>
```

Though the identifier is often formatted as 123-45-6789, the "-" should be removed so the right format is 123456789.

7.6.6.7.3 NHS Number

```
<example xsi:type="II" root="2.16.840.1.113883.2.1.4.1"
extension="9999999484" reliability="VER" scope="BUSN"/>
```

This is an example NHS number from England/Wales. The NHS root OID is 2.16.840.1.113883.2.1.4.1. Because the NHS number is reused in multiple instances of a clinical document for the same patient, and for many other records, it will generally not have scope = OBJ – except on the NHS master repository itself. Usually the scope will be BUSN. Most NHS systems maintain patient information interfaces with the master NHS patient registry, or with other systems that do, so the reliability in these cases is “VER”.

7.6.6.7.4 Australian Medicare Number

```
<example xsi:type="II" root="1.2.36.174030967" extension="1234567892"
reliability="VRF" scope="OBJ"/>
```

© ISO 2007 – All rights reserved

This is an Australian medicare number. The root is the business root oid for the issuing authority (HIC. This should be a further qualified oid, but they have not, as yet, published their own scheme for their oid space). The reliability is "VRF" because the sending system has checked that this is the correct identity for the patient with HIC itself using one of their electronic interfaces, and the scope is OBJ because this number is only used for identifying the account associated with this patient's family by the HIC.

7.6.6.7.5 Record Identifier

```
<example xsi:type="II" root="D6A7AB37-4220-4D80-9052-8A4959A203E3"
reliability="ISS" scope="VER"/>
```

A UUID issued by the sending system associated with the particular version of the record being represented in the instance.

7.6.6.7.6 Lab Number

```
<example xsi:type="II" root="2.16.840.1.113883.19.5.34"
extension="2345344" reliability="ISS" scope="OBJ"/>
```

A Laboratory identifier that identifies the report. This has scope=”OBJ” and will be retained throughout the lab workflow as the sample is received, processed, reported, and signed.

7.6.6.7.7 EN 13606 Record Component Id

```
<example xsi:type="II" root="2.16.840.1.113883.19.5.462"
extension="976295765" reliability="VRF" scope="VER"/>
```

ISO/EN 13606-1 rc_id (record component id), which is persisted across EHR repositories but not re-used across different versions of the record, and not-reused between workflow state changes. Hence the scope is "VER". The reliability is VRF; in the context of 13606 the use of ISS is not recommended as it can only be used by the primary issuing system, and this makes it infeasible to attest to the record.

The primary use of ISS is in patient management etc, where it can be used to help build linking/unlinking/merging workflows.

7.6.6.7.8 Message Snapshot

```
<example xsi:type="SET_II">
  <id root="2.16.840.1.113883.19.5.971" extension="763491"
     reliability="VRF" scope="OBJ"/>
  <id root="2.16.840.1.113883.19.5.972" extension="351324"
     reliability="VRF" scope="VER"/>
  <id root="0282CA34-2E4E-4B9D-82A5-BD2BF8497940"
     reliability="ISS" scope="VW"/>
</example>
```

This example shows the full use of a SET(II) in a message. The message carries a snapshot of an object in a class in the message, and the class has an attribute id : SET(II). The information in in the class that carries the SET(II) attribute is based on the object 763491 in the OID space 2.16.840.1.113883.19.5.971. The version the snapshot is based on is identified by the identifier 351324 in the OID space 2.16.840.1.113883.19.5.972. Finally, the snapshot is given it’s own UUID identifier 0282CA34-2E4E-4B9D-82A5-BD2BF8497940, which may be used if a system wished to record that particular snapshot it it’s audit trail.

© ISO 2007 – All rights reserved

7.7 Name and Address Datatypes

These data types provide support for names and addresses.

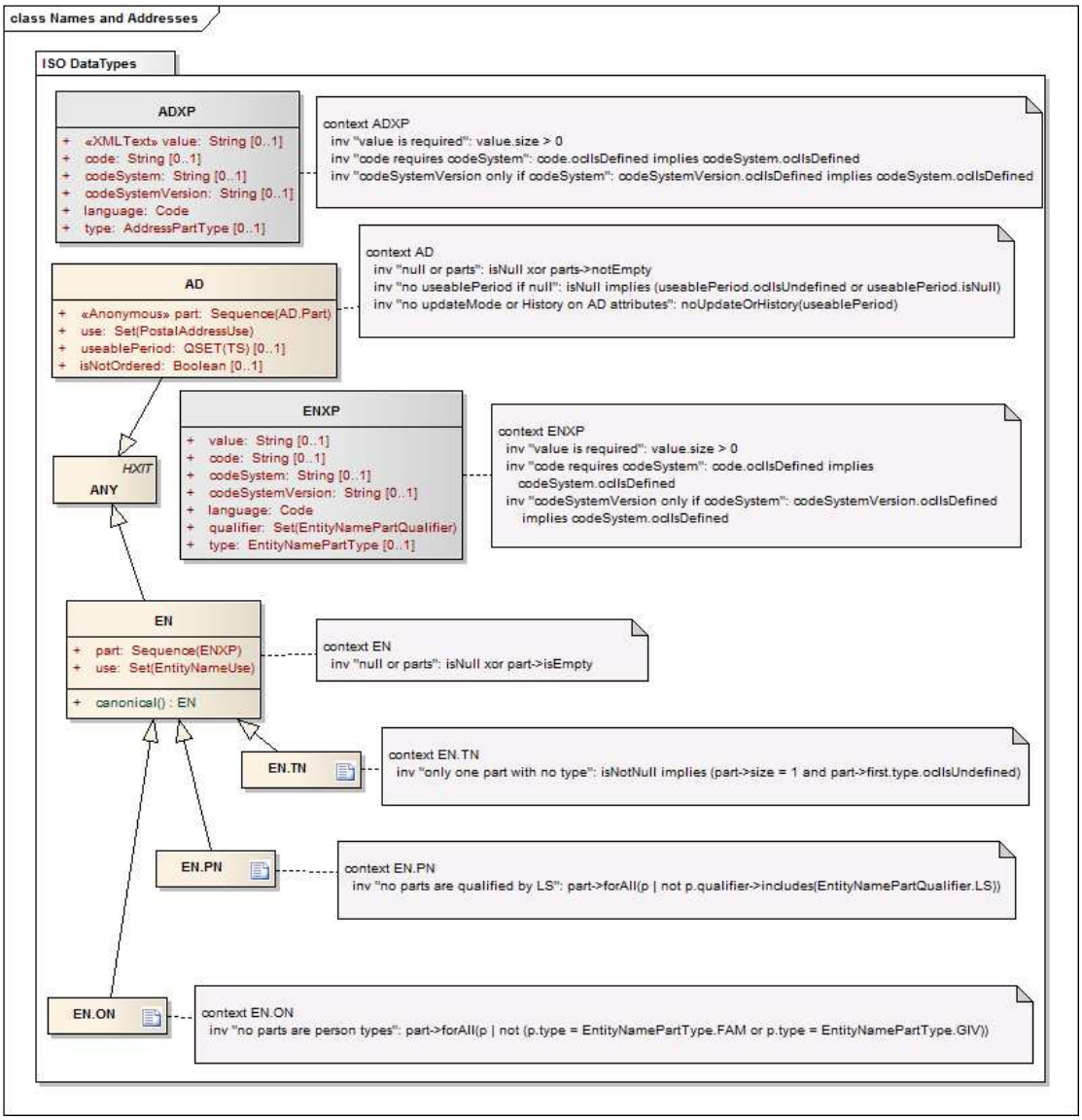


Figure 6. Name and address data types

7.7.1 ADXP (Address Part)

7.7.1.1 Description:

A character string that may have a type-tag signifying its role in the address. Typical parts that exist in about every address are street, house number, or post box, postal code, city, country but other roles may be defined regionally, nationally, or on an enterprise level (e.g. in military addresses).

Addresses are usually broken up into lines, which may be indicated by special line-breaking delimiter elements (e.g., DEL).

© ISO 2007 – All rights reserved

In addition to the string that represents the part, the part may also be encoded using the code property. The string content must always be provided whether the content is encoded or not.

7.7.1.2 ISO 11404 Syntax:

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

7.7.1.3 Attributes

7.7.1.3.1 value : String: The actual string value of the part.

7.7.1.3.2 code : String: A code assigned to the part by some coding system, if appropriate.

7.7.1.3.3 codeSystem : String: The code system from which the code is taken.

The choice of coding system depends on the part type. The type CNT (Country) is bound to the 3 letter codes defined in ISO 3166. The context of use may specify other bindings for other part types.
The codeSystem SHALL be populated if a code is populated.

7.7.1.3.4 codeSystemVersion : String: The version of the coding system, if required.

The codeSystem SHALL be populated if a codeSystemVersion is populated.

7.7.1.3.5 language: Code: The human language of the content. Valid codes are taken from the IETF RFC 3066. If this attribute is null, the language may be inferred from elsewhere, either from the context or from unicode language tags, for example.

Note: While values of type ADXP may be assigned a language, the meaning of the part is not dependent on the language, and applications SHALL not be required to indicate the linguistic origin of any address part..

7.7.1.3.6 type : AddressPartType: Specifies whether an address part names the street, city, country, postal code, post box, etc.

If the type is NULL the address part is unclassified and would simply appear on an address label as is.

If populated, the value of this attribute SHALL be taken from the HL7 AddressPartType code system. The current values are:

AddressPartType Enumeration. OID: 2.16.840.1.113883.5.16			
1	AL	address line	An address line is for either an additional locator, a delivery address or a street address. An address generally has only a delivery address line or a street address line, but not both
2	ADL	additional locator	This can be a unit designator, such as apartment number, suite number, or floor. There may be several unit designators

© ISO 2007 – All rights reserved

			in an address (e.g., "3rd floor, Appt. 342"). This can also be a designator pointing away from the location, rather than specifying a smaller location within some larger one (e.g., Dutch "t.o." means "opposite to" for house boats located across the street facing houses)
3	UNID	unit identifier	The number or name of a specific unit contained within a building or complex, as assigned by that building or complex
3	UNIT	unit designator	Indicates the type of specific unit contained within a building or complex. E.g. Apartment, Floor
2	DAL	delivery address line	A delivery address line is frequently used instead of breaking out delivery mode, delivery installation, etc. An address generally has only a delivery address line or a street address line, but not both
3	DINST	delivery installation type	Indicates the type of delivery installation (the facility to which the mail will be delivered prior to final shipping via the delivery mode.) Example: post office, letter carrier depot, community mail center, station, etc
3	DINSTA	delivery installation area	The location of the delivery installation, usually a town or city, and is only required if the area is different from the municipality. Area to which mail delivery service is provided from any postal facility or service such as an individual letter carrier, rural route, or postal route
3	DINSTQ	delivery installation qualifier	A number, letter or name identifying a delivery installation. E.g., for Station A, the delivery installation qualifier would be ‘A’
3	DMOD	delivery mode	Indicates the type of service offered, method of delivery. For example: post office box, rural route, general delivery, etc
3	DMODID	delivery mode identifier	Represents the routing information such as a letter carrier route number. It is the identifying number of the designator (the box number or rural route number)
2	SAL	street address line	A street address line is frequently used instead of breaking out build number, street name, street type, etc. An address generally has only a delivery address line or a street address line, but not both.
3	BNR	building number	The number of a building, house or lot alongside the street. Also known as "primary street number". This does not number the street but rather the building
4	BNN	building number numeric	The numeric portion of a building number
4	BNS	building number suffix	Any alphabetic character, fraction or other text that may appear after the numeric portion of a building number
3	STR	street name	The name of the street, including the type
4	STB	street name base	The base name of a roadway or artery recognized by a municipality (excluding street type and direction)
4	STTYP	street type	The designation given to the street. (e.g. Street, Avenue, Crescent, etc.)
3	DIR	direction	Direction (e.g., N, S, W, E)

© ISO 2007 – All rights reserved

2	INT	intersection	An intersection denotes that the actual address is located at the intersection or two or more streets
1	CAR	care of	The name of the party who will take receipt at the specified address, and will take on responsibility for ensuring delivery to the target recipient
1	CEN	census tract	A geographic sub-unit delineated for demographic purposes
1	CNT	country	Country
1	CPA	county or parish	A sub-unit of a state or province. (49 of the United States of America use the term "county;" Louisiana uses the term "parish")
1	CTY	municipality	The name of the city, town, village, or other community or delivery center
1	DEL	delimiter	Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break.
1	POB	post box	A numbered box located in a post station
1	PRE	precinct	A subsection of a municipality
1	STA	state or province	A sub-unit of a country with limited sovereignty in a federally organized country
1	ZIP	postal code	A postal code designating a region defined by the postal service

Note: The hierarchical nature of this code system shows composition rather than subsumption. E.g. "Street Name" is part of "Street Address Line"

7.7.1.4 ISO 11404 Syntax:

```
 type AddressPartType = enumeration (AL, ADL, UNID, UNIT,
DAL, DINST, DINSTA, DINSTQ, DMOD, DMODID, SAL, BNR, BNN,
BNS, STR, STB, STTYP, DIR, INT, CAR, CEN, CNT, CPA, CTY,
DEL, POB, PRE, STA, ZIP)
```

7.7.1.5 Equality

Two ADXP values are equal if their type and value attributes are equal. The code attributes and language are ignored.

Note (Clarification): two type attributes of null are considered equal.

7.7.1.6 Invariants

- The value cannot be empty
- if code has a value then codeSystem must have a value
- codeSystemVersion can only have a value if codeSystem has a value

OCL for Invariants:

```
inv "value is required": value.size > 0
inv "code requires codeSystem": code.oclIsDefined implies
```


© ISO 2007 – All rights reserved

```
    codeSystem.oclIsDefined
inv "codeSystemVersion only if codeSystem":
    codeSystemVersion.oclIsDefined implies
    codeSystem.oclIsDefined
```

7.7.2 AD.Part (Address)

Stereotype: «Choice»

Description: Content model for addresses. The choice stereotype denotes that exactly one of the attributes SHALL have a value. All the others must be null.

Rather than simply listing the parts of an address as simple parts with a type and value, addresses are structured as a sequence of named parts where the type is inferred from the name.

This UML diagram shows how the model is created:

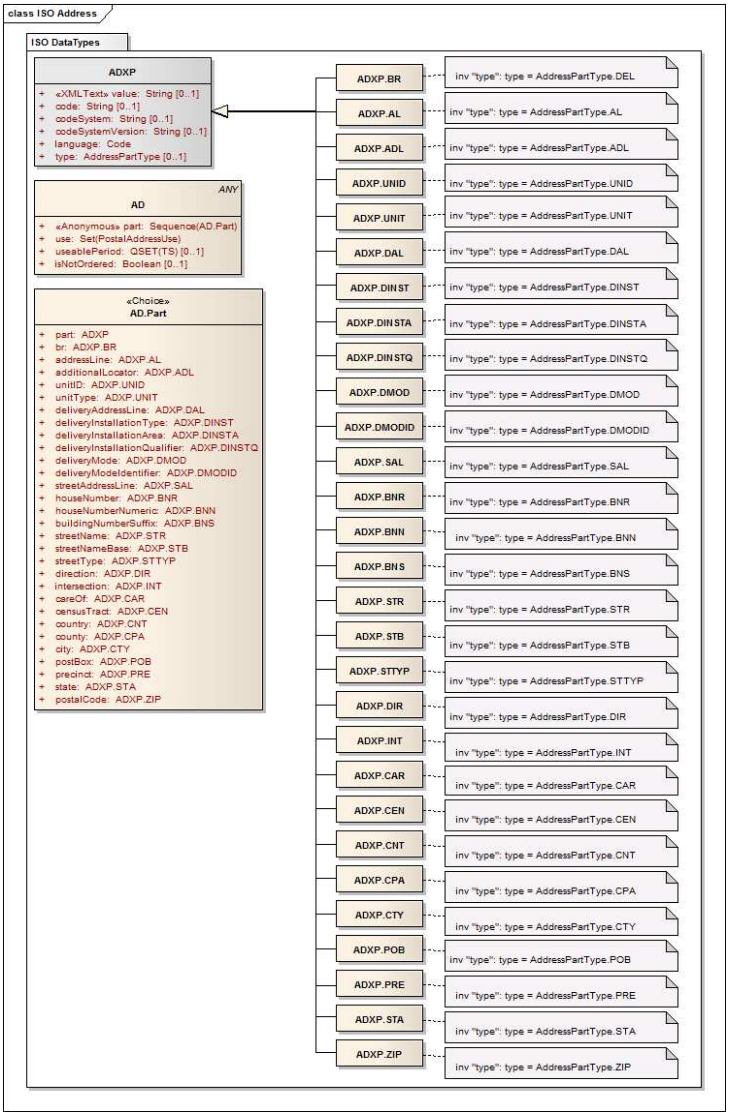


Figure 7. Address Content Model

© ISO 2007 – All rights reserved

Note that plain unadorned parts may still be used; these are required to send parts that have an unknown type, and parts that have a type code that is not defined at the time this standard is finalized. There is also a part called br that has no semantic significance but represents an explicit line break when the address is presented for human consumption.

7.7.3 AD (Address)

Specializes ANY

Description: Mailing and home or office addresses.

AD is primarily used to communicate data that will allow printing mail labels, or that will allow a person to physically visit that address. The postal address datatype is not supposed to be a container for additional information that might be useful for finding geographic locations (e.g., GPS coordinates) or for performing epidemiological studies. Such additional information should be captured by other, more appropriate data structures.

Addresses are essentially sequences of address parts, but add a "use" code and a valid time range for information about if and when the address can be used for a given purpose.

11404 Syntax:

```
type AD = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  part : Sequence(ADXP),
  use : Set(PostalAddressUse),
  useablePeriod : QSET(TS),
  isNotOrdered : boolean
)
```

7.7.3.1 Attributes

7.7.3.1.1 part : Sequence(AD.Part) «Anonymous»: A sequence of address parts, such as street or post office Box, city, postal code, country, etc.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.7.3.1.2 use : Set(PostalAddressUse): A set of codes advising a system or user which address in a set of like addresses to select for a given purpose.

An address without specific use code might be a default address useful for any purpose, but an address with a specific use code would be preferred for that respective purpose.

If populated, the values contained in this attribute SHALL be taken from the HL7 PostalAddressUse code system. The current values are:

PostalAddressUse Enumeration. OID: 2.16.840.1.113883.5.1012

© ISO 2007 – All rights reserved

1	*AddressUse*		
2	H	home address	A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available
3	HP	primary home	The primary home, to reach a person after business hours
3	HV	vacation home	A vacation home, to reach a person while on vacation
2	WP	work place	An office address. First choice for business related contacts during business hours
3	DIR	direct	Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries. For phones, often referred to as a 'private line'
3	PUB	public	Indicates a work place address or telecommunication address that is a 'standard' address which may reach a reception service, mail-room, or other intermediary prior to the target entity
2	BAD	bad address	A flag indicating that the address is bad, in fact, useless
2	PHYS	physical visit address	Used primarily to visit an address
2	PST	postal address	Used to send mail
2	TMP	temporary address	A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.
1	*AddressRepresentationUse.* Identifies the different representations of the address. The representation may affect how the address is used. (E.g. use of Ideographic for formal communications)		
2	ABC	alphabetic	Alphabetic transcription of name (Japanese: romaji)
2	IDE	ideographic	Ideographic representation of name (e.g., Japanese kanji, Chinese characters)
2	SYL	syllabic	Syllabic transcription of name (e.g., Japanese kana, Korean hangul)
1	SRCH	Search Type Uses	A name intended for use in searching or matching.
2	SNDX	soundex	An address spelled according to the SoundEx algorithm
2	PHON	phonetic	The address as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the address, not based on a phonetic algorithm

ISO 11404 Syntax for the postalAddressUse attribute

```
type PostalAddressUse = enumeration (H, HP, HV, WP, DIR,
PUB, BAD, TMP, ABC, IDE, SYL, PHYS, PST, SRCH, SNDX, PHON)
```

Note: Only one of the values ABC, IDE, and SYL may be used in the set. There is no constraints on the other combinations, though some of them may not make sense.

© ISO 2007 – All rights reserved

7.7.3.1.3 useablePeriod : QSET(TS): A General Timing Specification (GTS) specifying the periods of time during which the address can be used. This is used to specify different addresses for different times of the week or year.

7.7.3.1.4 isNotOrdered : Boolean: A boolean value specifying whether the order of the address parts is known or not. While the address parts are always a sequence, the order in which they are presented may or may not be known to be true or important. Where this matters, the isNotOrdered property can be used to convey this information.

7.7.3.2 Equality

Two address values are considered equal if both contain the same address parts, independent of ordering. Use code, useablePeriod, and isNotOrdered are excluded from the equality test.

Note 1: even if isNotOrdered is false – it is known that the order of the address parts is representationally significant – the order of the parts is irrelevant for checking equality of addresses.

Note 2: two values that refer to the same address but that are encoded using different address parts (perhaps to different levels of detail) would not be considered equal.

7.7.3.3 Invariants

- Either the AD is null or it has at least one part

OCL for Invariants:

```
        inv "null or parts": isNull xor part->notEmpty

nullflavor invariants:
        inv "no useablePeriod if null": isNull implies
          (useablePeriod.oclIsUndefined or
             useablePeriod.isNull)
        inv "no updateMode or History on AD attributes":
             noUpdateOrHistory(useablePeriod)
```

7.7.3.4 ISO 22220 Comments

The various address parts defined by ISO 22220 map to address part types, and the address type maps to the use attribute. The start and end date accuracy indicators are partially supported by the precision of the dates provided.

7.7.3.5 Examples

7.7.3.5.1 Address with Layout

```
<example xsi:type="AD" use="WP">
   <part>1050 W Wishard Blvd,</part>
   <br/>
   <part>RG 5th floor,</part>
   <br/>
   <part>Indianapolis, IN 46240</part>
```


© ISO 2007 – All rights reserved

```
</example>
```

This work address consists of 3 unknown parts with 2 line delimiters. None of the parts are labelled in regard to their semantic significance.

7.7.3.5.2 Address with Types

```
<example xsi:type="AD" use="WP">
   <addressLine>1050 W Wishard Blvd</addressLine>
   <addressLine>RG 5th floor</addressLine>
   <city>Indianapolis</city>
   <state>IN</state>
   <postalCode>46240</postalCode>
</example>
```

This is the same address using standard typing rather than a presentation focus. This is probably be the most common form of presentation for addresses - a series of address lines followed by city, state, zip and possibly country.

Note: Although this presentation of the address suggests that lines are required after the two address lines, this is not implied by this example. See Section 7.7.3.6.

7.7.3.5.3 Line Types

```
<example xsi:type="AD" use="WP">
   <streetAddressLine>1050 W Wishard Blvd</streetAddressLine>
   <additionalLocator>RG 5th floor</additionalLocator>
   <city>Indianapolis</city>
   <state>IN</state>
   <postalCode>46240</postalCode>
</example>
```

This is the same address from a system that differentiates between different line types .

7.7.3.5.4 Fully Typed Addresses

```
<example xsi:type="AD" use="WP">
   <houseNumber>1050</houseNumber>
   <direction>W</direction>
   <streetNameBase>Wishard</streetNameBase>
   <streetType>Blvd</streetType>
   <additionalLocator>RG 5th floor</additionalLocator>
   <city>Indianapolis</city>
   <state>IN</state>
   <postalCode>46240</postalCode>
</example>
```

The same address fully broken down. The form above is not used in the USA. However, it is useful in Germany, where many systems keep house number as a distinct field

```
<example xsi:type="AD" use="HP">
   <streetName>Windsteiner Weg</streetName>
   <houseNumber>54a</houseNumber>
   <country code="DEU"
       codeSystem="2.16.840.1.113883.11.171"/>D</country>
   <postalCode>14165</postalCode>
   <city> Berlin </city>
</example>
```

This is a home address in a standard German format. The country has been coded in ISO 3166-3 to assist with interoperability.

© ISO 2007 – All rights reserved

7.7.3.5.5 Unknown Addresses

```
<example xsi:type="AD" use="WP" nullFlavor="UNK"/>
```

The work address is unknown.

7.7.3.6 Presenting Addresses

The primary purpose of an address is to be presented on a delivery label affixed to some envelope. A fully specified address - one that includes specified line breaks - can be presented directly by simply presenting the text of the various parts with whitespace separating them, and following the explicit line breaks. If the elements are moved into the xhtml namespace, the AD content can be treated as html directly.

For this reason, the address SHOULD always be generated with appropriate line breaks included in the address. This enables applications that do not understand the semantics of the address to reproduce it correctly.

However because there is no single presentation model for addresses, applications may ignore the explicitly specified line breaks in addresses – they are not bound to follow the presentation as specified in any particular address.

7.7.4 ENXP (Entity Name Part)

7.7.4.1 Description: A character string token representing a part of a name.

May have a type code signifying the role of the part in the whole entity name, and qualifier codes for more detail about the name part type. (Typical name parts for person names are given names, and family names, titles, etc.). In addition to the string that represents the part, the part may also be encoded using the code property.

The string content must always be provided whether the content is encoded or not.

11404 Syntax:

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


© ISO 2007 – All rights reserved

7.7.4.2 Attributes

7.7.4.2.1 value : String: The actual string value of the part.

7.7.4.2.2 code : String: A code assigned to the part by some coding system, if appropriate.

7.7.4.2.3 codeSystem : String: The code system from which the code is taken. The choice of coding system depends on the part type.

The codeSystem SHALL be populated if a code is populated.

7.7.4.2.4 codeSystemVersion : String: The version of the coding system, if required.

The codeSystem SHALL be populated if a codeSystemVersion is populated.

7.7.4.2.5 language: Code: The human language of the content. Valid codes are taken from the IETF RFC 3066. If this attribute is null, the language may be inferred from elsewhere, either from the context or from unicode language tags, for example.

Note: While values of type ENXP may be assigned a language, the meaning of the part is not dependent on the language, and applications SHALL not be required to indicate the linguistic origin of any name part..

7.7.4.2.6 type : EntityNamePartType: Indicates whether the name part is a given name, family name, prefix, suffix, etc.

Note: Not every name part must have a type code, if the type code is unknown, not applicable, or simply undefined this is expressed by a NULL value (type.isNull). For example, a name may be "Rogan Sulma" and it may not be clear which one is a given name or which is a last name, or whether Rogan may be a title.

If populated, the value of this attribute SHALL be taken from the HL7 EntityNamePartType code system. The current values are:

EntityNamePartType Enumeration. OID: 2.16.840.1.113883.5.44			
1	FAM	family	Family name, this is the name that links to the genealogy. In some cultures (e.g. Eritrea) the family name of a son is the first name of his father
1	GIV	given	Given name. Note: don't call it "first name" since this given names do not always come first
1	PFX	prefix	A prefix has a strong association to the immediately following name part. A prefix has no implicit trailing white space (it has implicit leading white space though). Note that prefixes can be inverted
1	SFX	suffix	A suffix has a strong association to the immediately preceding name part. A suffix has no implicit leading white space (it has implicit trailing white space though). Suffices can not be inverted
1	DEL	delimiter	A delimiter has no meaning other than being literally printed in this name representation. A delimiter has no implicit leading and trailing white space

© ISO 2007 – All rights reserved

ISO 11404 Syntax for the entityNamePartType attribute

```
 type EntityNamePartType = enumeration (FAM, GIV, PFX, SFX,
DEL)
```

When a name is hyphenated, such as Mary-Ann, it may be ambigious whether to use a delimiter separating two name parts, or a single name part with a hyphen in it. As a rule of thumb, if each name part should contribute an initial when the name is presented as initials, then a delimiter should be used to separate two parts.

7.7.4.2.7 qualifier : Set(EntityNamePartQualifier): The qualifier is a set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type.

Example, a given name may be flagged as a nickname (CL), a family name may be a name acquired by marriage (SP) or a name from birth (BR).

If populated, the values contained in this attribute SHALL be taken from the HL7 EntityNamePartQualifier code system. The current values are:

EntityNamePartQualifier Enumeration. OID: 2.16.840.1.113883.5.43			
1	LS	legal status	For organizations a suffix indicating the legal status, e.g., "Inc.", "Co.", "AG", "GmbH", "B.V." "S.A.", "Ltd." Etc.
1	AC	academic	Indicates that a prefix like "Dr." or a suffix like "M.D." or "Ph.D." is an academic title
1	NB	nobility	In Europe and Asia, there are still people with nobility titles (aristocrats). German "von" is generally a nobility title, not a mere voorvoegsel. Others are "Earl of" or "His Majesty King of..." etc. Rarely used nowadays, but some systems do keep track of this
1	PR	professional	Primarily in the British Imperial culture people tend to have an abbreviation of their professional organization as part of their credential suffices
1	VV	voorvoegsel	A Dutch "voorvoegsel" is something like "van" or "de" that might have indicated nobility in the past but no longer so. Similar prefixes exist in other languages such as Spanish, French or Portugese, etc
1	AD	adopted	The name the person was given at the time of adoption
1	BR	birth	A name that a person had shortly after being born. Usually for family names but may be used to mark given names at birth that may have changed later
1	SP	spouse	The name assumed from the partner in a marital relationship. Usually the spouse's family name. Note that no inference about gender can be made from the existence of spouse names
1	CL	callme	A callme name is usually a given name that is preferred when a person is directly addressed. Sometimes known as a nickname
1	IN	initial	Indicates that a name part is just an initial. Initials do not imply a trailing period since this would not work with non-Latin scripts. Initials may consist of more than one letter, e.g., "Ph." could stand for "Philippe" or "Th." for "Thomas"
1	TITLE	title	Indicates that a prefix or a suffix is a title that applies to the

© ISO 2007 – All rights reserved

			whole name, not just the adjacent name part

ISO 11404 Syntax for the entityNamePartQualifier attribute

```
 type EntityNamePartQualifier = enumeration (LS, AC, NB,
PR, VV, AD, BR, SP, CL, IN, TITLE)
```

7.7.4.3 Equality

Two ENXP values are equal if their type and value attributes are equal. The code attributes, language and qualifier are ignored.

Note (Clarification): two type attributes of null are considered equal.

7.7.4.4 Invariants

- The value cannot be empty
- if code has a value then codeSystem must have a value
- codeSystemVersion can only have a value if codeSystem has a value

OCL for Invariants:

```
inv "value is required": value.size > 0
inv "code requires codeSystem": code.oclIsDefined
   implies codeSystem.oclIsDefined
inv "codeSystemVersion only if codeSystem":
   codeSystemVersion.oclIsDefined implies
    codeSystem.oclIsDefined
```

7.7.4.5 Implementation Notes

There is a relationship between the part type and the qualifiers that can be used. This table summarises the qualifiers that can be used with the different part types:

	FAM (Family)	GIV (Given)	PFX (Prefix)	SFX (Suffix)	DEL (Delimiter)
LS (legal status)			✓	✓✓	
AC (academic)			✓✓	✓✓	
NB (nobility)			✓✓		
PR (professional)			✓✓	✓✓	
VV (voorvoegsel)			✓✓		
AD (adopted)	✓✓	✓✓	✓✓	✓✓	
BR (birth)	✓✓	✓✓	✓✓	✓✓	
SP (spouse)	✓✓	✓	✓	✓	
CL (call me)	✓	✓✓	✓	✓	
IN (initial)		✓✓			
TITLE (title)			✓✓	✓✓	

✓ = This combination is allowed.

© ISO 2007 – All rights reserved

✓✓ = This combination is allowed, and it is expected that this combination will be encountered in practice.

7.7.5 EN (Entity Name)

7.7.5.1 Description

Specializes ANY

A name for a person, organization, place or thing.

Examples: "Jim Bob Walton, Jr.", "Health Level Seven, Inc.", "Lake Tahoe", etc. An entity name may be as simple as a character string or may consist of several entity name parts, such as, "Jim", "Bob", "Walton", and "Jr.", "Health Level Seven" and "Inc.".

Entity names are essentially sequences of entity name parts, but add a "use" code and a valid time range for information about when the name was used and how to choose between multiple aliases that may be valid at the same point in time.

7.7.5.2 ISO 11404 Syntax:

```
type EN = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  part : Sequence(ENXP),
  use : Set(EntityNameUse
)
```

7.7.5.3 Attributes

7.7.5.3.1 part : Sequence(ENXP): A sequence of name parts, such as given name or family name, prefix, suffix, etc.

7.7.5.3.2 use : Set(EntityNameUse): A set of codes advising a system or user which name in a set of names to select for a given purpose.

A name without specific use code might be a default name useful for any purpose, but a name with a specific use code would be preferred for that respective purpose.

If populated, the values contained in this attribute SHALL be taken from the HL7 EntityNameUse code system. The current values are:

EntityNameUse Enumeration. OID: 2.16.840.1.113883.5.45			
1	*RepresentationUse.* Identifies the different representations of a name. The representation may affect how the name is used. (E.g. use of Ideographic for formal communications)		
2	ABC	alphabetic	Alphabetic transcription of name (Japanese: romaji)
2	IDE	ideographic	Ideographic representation of name (e.g., Japanese kanji, Chinese characters)

© ISO 2007 – All rights reserved

2	SYL	syllabic	Syllabic transcription of name (e.g., Japanese kana, Korean hangul)
1	C	license	As recorded on a license, record, certificate, etc. (only if different from legal name)
1	L	legal	Known as/conventional/the one you use
1	ASGN	assigned	A name assigned to a person. Reasons some organizations assign alternate names may include not knowing the person's name, or to maintain anonymity. Some, but not necessarily all, of the name types that people call "alias" may fit into this category
1	I	indigenous/ tribal	e.g. Chief Red Cloud
1	P	pseudonym	A self asserted name that the person is using or has used
2	A	artist/stage	Includes writer's pseudonym, stage name, etc
1	R	religious	e.g. Sister Mary Francis, Brother John
1	OR	official record	the formal name as registered in an official (government) registry, but which name might not be commonly used. Particularly used in countries with a law system based on Napoleonic law
1	SRCH	Search Type Uses	A name intended for use in searching or matching
2	SNDX	soundex	An name spelled according to the SoundEx algorithm
2	PHON	phonetic	The name as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the address, not based on a phonetic algorithm.

ISO 11404 Syntax for the entityNameUse attribute

```
 type EntityNameUse = enumeration (C, I, L, P, A, R, OR,
SRCH, PHON, SNDX, ABC, SYL, IDE, ASGN)
```

Note: Only one of the values ABC, IDE, and SYL may be used in the set. There is no constraints on the other combinations, though some of them may not make sense.

7.7.5.4 Equality

Two name values are considered equal if their canonical forms both contain the same name parts in the same order. Use code and valid time are excluded from the equality test.

7.7.5.5 Invariants

- Either the EN is null or it has at least one part

OCL for Invariants:

```
inv "null or parts": isNull xor part-> notEmpty
```

© ISO 2007 – All rights reserved

7.7.5.6 Operations

7.7.5.6.1 **canonical():EN:** The Entity Name with a standard ordering imposed on the parts.

The canonical form is primarily defined for the purposes of defining equality, and may differ from the socially accepted order for the name parts in various cultures around the world.

The canonical form contains all the part types except for the delimiters, in the following order:

1. prefixes with qualifier title
2. given names, with any prefixes and/or suffixes associated with the given names
3. family names, with any prefixes and/or suffixes associated with the given names
4. suffixes with qualifier title

Each list of part types SHALL be in the order of the original name.

7.7.5.7 ISO 22220 Comments

The various name groups defined in ISO 22220 map directly to the ENXP types. The conditional use and Name Usage components map to the use attribute. The context of use of the EN may need to allow for multiple EN values (as some kind of collection) to support all the functionality described in ISO 22220.

7.7.5.8 Examples

7.7.5.8.1 Simple Example

```
<example xsi:type="EN" >
   <part type="GIV" value="Adam"/>
   <part type="GIV" value="A."/>
   <part type="FAM" value="Everyman"/>
</example>
```

A very simple encoding of "Adam A. Everyman".

7.7.5.8.2 Complex Example

```
<example xsi:type="EN.PN">
   <part type="GIV" qualifier="AC" value="Dr. phil."/>
   <part type="GIV" value="Regina"/>
   <part type="GIV" value="Johanna"/>
   <part type="GIV" value="Maria"/>
   <part type="PFX" qualifier="NB" value="Graufin"/>
   <part type="FAM" qualifier="BR" value="Hochheim"/>
   <part type="DEL" value="-"/>
   <part type="FAM" qualifier="SP" value="Weilenfels"/>
   <part type="SFX" qualifier="PR" value="NCFSA" />
</example>
```

This example shows extensive use of multiple given names, prefixes, suffixes, for academic degrees, nobility titles, and professional designations.

© ISO 2007 – All rights reserved

7.7.5.8.3 Organization Name

```
<example xsi:type="EN.TN">
   <part value="Health Level Seven, Inc"/>
</example>
```

An organization name, "Health Level Seven, Inc." in simple string form: (EN.TN).

```
<example xsi:type="EN.ON">
   <part value="Health Level Seven, "/>
   <part type="SFX" qualifier="LS" value="Inc."/>
</example>
```

As a fully parsed name.

7.7.5.8.4 Japanese Example

```
<example xsi:type="EN" use="IDE">
   <part type="FAM" value="木村"/>
   <part type="GIV" value="通男"/>
</example>
<example xsi:type="EN" use="SYL">
   <part type="FAM" value="きむら"/>
   <part type="GIV" value="みちお"/>
</example>
<example xsi:type="EN" use="ABC">
   <part type="FAM" value="KIMURA"/>
   <part type="GIV" value="MICHIO"/>
</example>
```

A Japanese name in the three forms: ideographic (Kanji), syllabic (Hiragana), and alphabetic (Romaji).

7.7.6 EN.TN (Trivial Name)

7.7.6.1 Description

Constrains EN

A restriction of EN that is effectively a simple string used for a simple name for things and places. Trivial names are typically used for places and things, such as Lake Erie or Washington-Reagan National Airport.

7.7.6.2 Invariants

- If the EN.TN is not null, there can only be one part, and it can have no type

OCL for Invariants:

```
inv "only one part with no type": isNotNull implies
     (part->size = 1 and part->first.type.oclIsUndefined)
```

© ISO 2007 – All rights reserved

7.7.7 EN.PN (Person Name)

7.7.7.1 Descripion

Constrains EN

A restriction of EN used when the named Entity is a Person. A sequence of name parts, such as given name or family name, prefix, suffix, etc.

A name part is a restriction on entity name part that only allows those entity name parts qualifiers applicable to person names.

Note: Since the structure of entity name is mostly determined by the requirements of person name, the restriction is very minor.

7.7.7.2 Invariants

- None of the parts of a persons name can be qualified by the status LS

OCL for Invariants:

```
inv "no parts are qualified by LS": part->forAll(p | not
       p.qualifier->includes(EntityNamePartQualifier.LS))
```

7.7.8 EN.ON (Organization Name)

Constrains EN

7.7.8.1 Invariants

- None of the parts of a organisation name can be FAM or GIV

OCL for Invariants:

```
inv "no parts are person types": part->forAll(p |
     not (p.type = EntityNamePartType.FAM or p.type =
      EntityNamePartType.GIV))
```


© ISO 2007 – All rights reserved

7.8 Quantity Datatypes

These data types provide support for Quantitative values

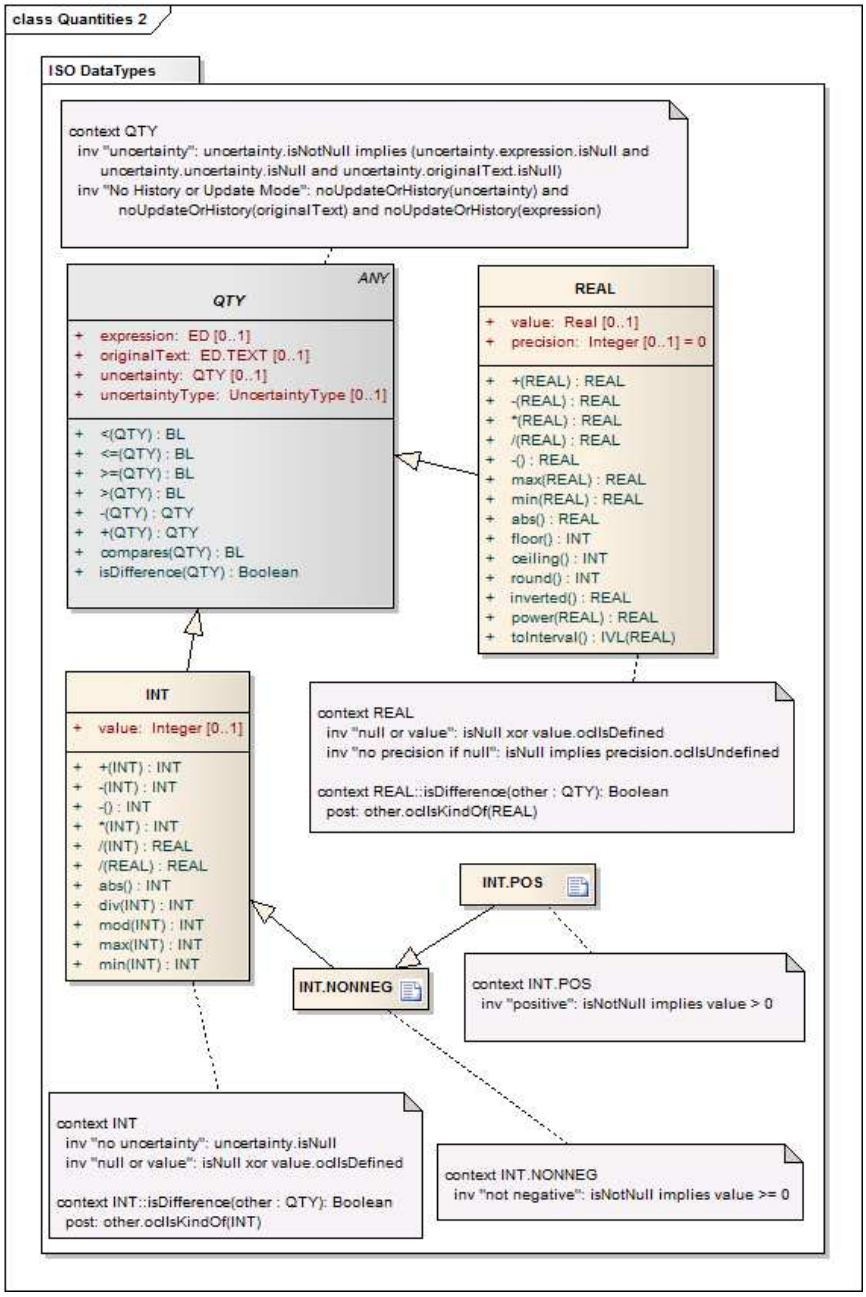


Figure 8. Quantity data types

© ISO 2007 – All rights reserved

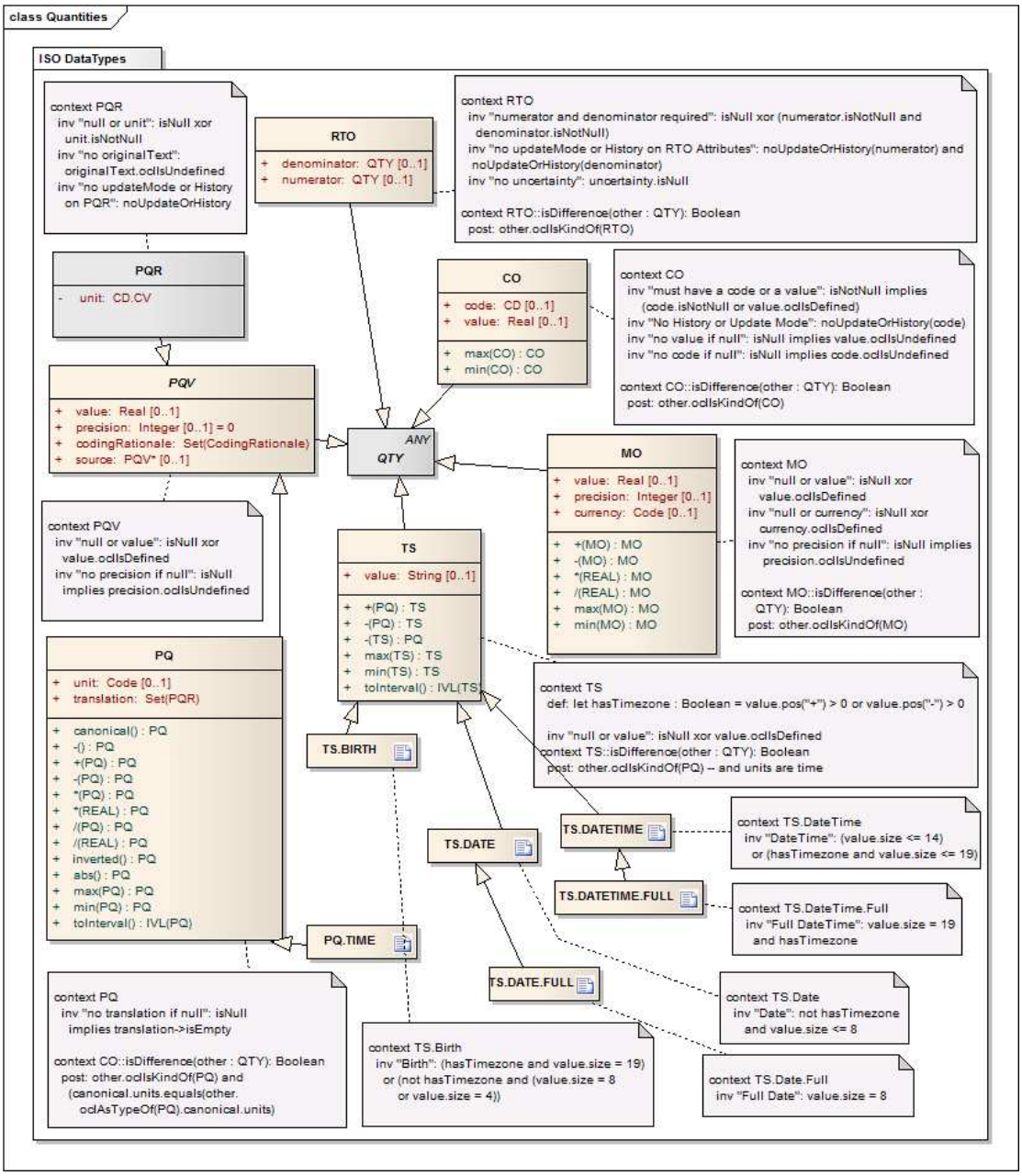


Figure 9. Quantity data types - continued

7.8.1 QTY (Quantity)

7.8.1.1 Description

Abstract. Specializes ANY

The quantity datatype is an abstract generalization for all datatypes whose domain values has an order relation (less-or-equal) and where difference is defined in all of the datatype's totally ordered value subsets.

The quantity type abstraction is needed in defining certain other types, such as the interval, and probability distributions.

© ISO 2007 – All rights reserved

7.8.1.2 11404 Syntax

```
type QTY = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType
)
```

The appropriate use of all of the attributes on QTY is intimately bound to the specification with which they are used. Generally the specification will need to establish special ways to control their use.

7.8.1.3 Attributes

7.8.1.3.1 expression : ED: An expression that can be used to derive the actual value of the quantity given information taken from the context of use.

For example expression can be used for expressing dosage instructions that depend on patient body weight. Note: If no proper value is provided for the QTY, then the value SHALL have a nullFlavor, whether or not an expression is provided.
The language of the expression is inferred from the mediatype. If multiple translations are provided in the expression, the evaluator is free to choose whichever language is preferred; all translations SHALL specify the same outcome.

The language defines the forms that the expression property can take, how the information available in the context of the expression is made available within the features of the language, and how the language declares the new form of the value. Languages may only be used if this information has been appropriately defined for the context in which the QTY is used.

Information Processing Entities are not required to implement any languages in order to claim direct or indirect conformance to this specification, but SHOULD declare what languages are supported in their conformance statements.

Language	Mediatype
OCL	text/plain+ocl
Factor	application/hl7-factor+xml
MathML	application/mathml+xml

Note: Factor is an HL7 specific language documented in the Abstract Datatypes Specification.

© ISO 2007 – All rights reserved

7.8.1.3.2 originalText : ED.TEXT: The text representation from which the QTY was encoded, if such a representation is the source of the QTY.

Original text can be used in a structured user interface to capture what the user saw as a representation of the quantity on the data input screen, or in a situation where the user dictates or directly enters text, it is the text entered or uttered by the user.

It is valid to use a QTY derived data type to store only the text that the user entered or uttered. In this situation, original text will exist without a valid value. In a situation where the value is determined sometime after the text was entered, originalText is the text or phrase used as the basis for determining the value. The originalText is not a substitute for a valid value. If the actual value of the QTY is not valid, then the QTY SHALL be null, irrespective of whether originalText has a value or not.

Notes

- The details of the link in the originalText.reference between different artifacts of medical information (e.g., document and coded result) is outside the scope of this specification and may be further proscribed in specifications that use this specification.

- The original text SHALL be an excerpt of the relevant information in the original sources, rather than a pointer or exact reproduction. Thus the original text SHALL be represented in plain text form. In specific circumstances, when clearly descirbed the context of use, the originalText may be a reference to some other text artefact for which the resolution scope is clearly described.

7.8.1.3.3 uncertainty : QTY: Specifies the uncertainty of the quantity using a distribution function and its parameters. It is the primary measure of variance/uncertainty of the value (the square root of the sum of the squares of the differences between all data points and the mean). The actual type of uncertainty depends on the type of the QTY.

Note: uncertainty SHALL only be applied to value domains that have a continuous distribution (REAL, PQ, MO, and TS). Uncertainty MAY be applied to the numerator and denominator of a RTO separately.

Uncertainty SHALL not have an expression. Uncertainty SHALL not have uncertainty of it's own. Uncertainty SHALL not have originalText.

7.8.1.3.4 uncertaintyType : UncertaintyType: A code specifying the type of probability distribution in uncertainty.

The NULL value (unknown) for the type code indicates that the probability distribution type is unknown. In that case, uncertainty has the meaning of an informal guess if it is populated.
If populated, the value of this attribute SHALL be taken from the HL7 DistributionType code system. The current values are:

UncertaintyType Enumeration. OID: 2.16.840.1.113883.5.1020			
1	U	uniform	The uniform distribution assigns a constant probability over the entire interval of possible outcomes, while all outcomes outside this interval are assumed to have zero probability. The width of this interval is $2\ \sigma\ \sqrt{3}$. Thus, the uniform distribution assigns the probability densities $f(x) = (2\ \sigma\ \sqrt{3})^{-1}$ to values $\mu - \sigma\ \sqrt{3} \geq x \leq \mu + \sigma\ \sqrt{3}$ and $f(x) = 0$ otherwise
1	N	normal (gaussian)	This is the well-known bell-shaped normal distribution. Because of the central limit theorem, the normal distribution is the distribution of choice for an unbounded random variable that is an outcome of a combination of many stochastic processes. Even for values bounded on a single side

© ISO 2007 – All rights reserved

			(i.e. greater than 0) the normal distribution may be accurate enough if the mean is "far away" from the bound of the scale measured in terms of standard deviations
1	LN	log-normal	The logarithmic normal distribution is used to transform skewed random variable X into a normally distributed random variable *U* = log *X*. The log-normal distribution can be specified with the properties mean μ and standard deviation σ. Note however that mean μ and standard deviation σ are the parameters of the raw value distribution, not the transformed parameters of the lognormal distribution that are conventionally referred to by the same letters. Those log-normal parameters μ_{log} and σ_{log} relate to the mean μ and standard deviation σ of the data value through $\sigma_{log}^2 = \log(\sigma^2/\mu^2 + 1)$ and $\mu\log = \log \mu - \sigma_{log}^2/2$
1	G	? (gamma)	The gamma-distribution used for data that is skewed and bounded to the right, i.e. where the maximum of the distribution curve is located near the origin. The γ-distribution has two parameters α and β. The relationship to mean μ and variance σ^2 is $\mu = \alpha\,\beta$ and $\sigma^2 = \alpha\,\beta^2$
1	E	exponential	Used for data that describes extinction. The exponential distribution is a special form of γ-distribution where α = 1, hence, the relationship to mean μ and variance σ^2 are $\mu = \beta$ and $\sigma^2 = \beta^2$
1	X2	?	Used to describe the sum of squares of random variables that occurs when a variance is estimated (rather than presumed) from the sample. The only parameter of the χ^2-distribution is υ, so called the *number of degrees of freedom* (which is the number of independent parts in the sum). The χ^2-distribution is a special type of γ-distribution with parameter $\alpha = \upsilon/2$ and $\beta = 2$. Hence, $\mu = \upsilon$ and $\sigma^2 = 2\,\upsilon$
1	T	t (student)	Used to describe the quotient of a normal random variable and the square root of a χ^2 random variable. The *t*-distribution has one parameter υ, the number of degrees of freedom. The relationship to mean μ and variance σ^2 are: $\mu = 0$ and $\sigma^2 = \upsilon / (\upsilon - 2)$
1	F	f	Used to describe the quotient of two χ^2 random variables. The *F*-distribution has two parameters υ_1 and υ_2, which are the numbers of degrees of freedom of the numerator and denominator variable respectively. The relationship to mean μ and variance σ^2 are: $\mu = \upsilon_2 / (\upsilon_2 - 2)$ and $\sigma^2 = (2\,\upsilon_2^2\,(\upsilon_2 + \upsilon_1 - 2)) / (\upsilon_1\,(\upsilon_2 - 2)^2\,(\upsilon_2 - 4))$
1	B	? (beta)	The beta-distribution is used for data that is bounded on both sides and may or may not be skewed (e.g., occurs when probabilities are estimated.) Two parameters α and β are available to adjust the curve. The mean μ and variance σ^2 relate as follows: $\mu = \alpha / (\alpha + \beta)$ and $(\sigma^2 = \alpha\,\beta/((\alpha + \beta)^2\,(\alpha + \beta + 1))$

Many distribution types are defined in terms of special parameters (e.g., the parameters α and β for the γ-distribution, number of degrees of freedom for the t-distribution, etc.). For all distribution types, however, the mean and standard deviation are defined.

If no value (null) is provided for distributionType, then the mean is estimated without any closer consideration of its probability distribution. In this case, the meaning of the standard deviation is not crisply defined. However, interpretation should be along the lines of the normal distribution, e.g., the interval covered by the mean ±1 standard deviation should be at the level of about two thirds confidence.

The three distribution-types *unknown* (null), *uniform* and *normal* must be supported by every system that claims to support uncertainty. All other distribution types are optional. When a

© ISO 2007 – All rights reserved

system interpreting a uncertainty representation encounters a distribution type that it does not recognize, it maps this type to the unknown (bull) distribution-type.

11404 Syntax for the distributionType attribute

```
 type UncertaintyType = enumeration (U, N, LN, G, E, X2, T,
F, B)
```

7.8.1.4 Equality

Equality is not defined for the QTY datatype as it is an abstract type.The QTY attributes (expression, originalText, uncertainty and uncertaintyType) never participate in the determination of equality of specializations of QTY.

7.8.1.5 Invariants

- Uncertainty has no expression, uncertainty, or originalText

OCL for Invariants:

```
inv "uncertainty": uncertainty.isNotNull implies
  (uncertainty.expression.isNull and
   uncertainty.uncertainty.isNull and
   uncertainty.originalText.isNull)
-- nullFlavor related
inv "No History or Update Mode":
     noUpdateOrHistory(uncertainty) and
       noUpdateOrHistory(originalText) and
       noUpdateOrHistory(expression)
```

7.8.1.6 Operations

7.8.1.6.1 lessthan[<](other : QTY):BL: True if the value of this is less than the value of other.

7.8.1.6.2 lessOrEqual[<=](other : QTY):BL: True if the value of this is less than or equal to the value of other.

7.8.1.6.3 greaterOrEqual[>=](other : QTY):BL: True if the value of this is greater than or equal to the value of other.

7.8.1.6.4 greaterThan[>](other : QTY):BL: True if the value of this is greater than the value of other.

7.8.1.6.5 plus[+](other : QTY):QTY: The result of addition of this and other. Other must be of the right type of value (same type as this, except for TS, where the value must be a PQ with units of time, or for PQ, where the value of other must have compatible units), else the result is null (nullFlavor NI).

7.8.1.6.6 minus[-](other : QTY):QTY: The result of subtraction of other from this. Other must be of the type of value (same type as this, except for TS, where the value must be a TS

© ISO 2007 – All rights reserved

or a PQ with units of time, or for PQ, where the value of other must have compatible units), else the result is null (nullFlavor NI).

7.8.1.6.7 compares(other : QTY):BL: whether this and other can be compared using equality.

Note: Generally this is true if both this and other are the same type unless noted otherwise for specializations for QTY.

7.8.1.6.8 isDifference(other : QTY):BL True if other is an instance that expresses the difference between two instances of this type.

Note: Usually this is true if other the same as the type, except for TS, where the difference is expressed as a PQ with a unit that is a kind of time, and for PQ, where the units must be compatible.

7.8.2 INT (Integer)

7.8.2.1 Description

Specializes QTY

Integer numbers (-1,0,1,2, 100, 3398129, etc.) are precise numbers that are results of counting and enumerating. Integer numbers are discrete, the set of integers is infinite but countable. No arbitrary limit is imposed on the range of integer numbers.

7.8.2.2 ISO 11404 Syntax:

```
type INT = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  value : integer
)
```

7.8.2.3 Attributes

value : Integer: The value of the INT if not null. Note that this specification imposes no limitations on the size of integer, but most implementations will map this to a 32 or 64 bit integer.

7.8.2.4 Equality

Two nonNull INT are equal if they are not null and have the same value.

© ISO 2007 – All rights reserved

7.8.2.5 Invariants

- a value must be provided if not null
- Uncertainty SHALL not be populated

OCL for Invariants:

```
inv “no uncertainty”: uncertainty.isNull
inv "null or value": isNull xor value.oclIsDefined
```

7.8.2.6 Operations

7.8.2.6.1 negated[-] : INT: The negative value of this.

7.8.2.6.2 plus[+] (other : INT) : INT: The value of the addition of this and other.

7.8.2.6.3 minus[-] (other : INT) : INT: The value of the subtraction of other from this.

7.8.2.6.4 times[*] (other : INT) : INT: The value of the multiplication of this and other.

7.8.2.6.5 dividedBy[/] (other : INT) : REAL: The value of this divided by other. If other is 0, then the result is nullFlavor NI.

7.8.2.6.6 dividedBy[/] (other : REAL) : REAL: The value of this divided by other. If other is 0, then the result is nullFlavor NI.

7.8.2.6.7 abs() : INT: The absolute value of this.

7.8.2.6.8 div(other : INT) : INT: The number of times that other fits completely within this.

7.8.2.6.9 mod(other : INT) : INT: The result is this modulo other.

7.8.2.6.10 max(other : INT) : INT: The maximum of this and other.

7.8.2.6.11 min(other : INT) : INT: The minimum of this and other.

7.8.2.7 Examples

7.8.2.7.1 Plain Value

```
<example xsi:type="INT" value="23"/>
```

The integer 23.

7.8.2.7.2 Unknown Value

```
<example xsi:type="INT" nullFlavor="NASK"/>
```

The patient wasn't asked for this value. For instance, the patient has never been pregnant, so that patient was not asked how many children she has.

© ISO 2007 – All rights reserved

7.8.3 INT.NONNEG

7.8.3.1 Description

Constrains INT

NT.NONNEG constrains INT so that it has a value of 0 or greater.

7.8.3.2 Invariants

- The value must be zero or greater if not null.

OCL for Invariants:

```
inv "not negative": isNotNull implies value >= 0
```

7.8.4 INT.POS

7.8.4.1 Description

Constrains INT.NONNEG

INT.POS constrains INT.NONNEG so that it has a value greater than 0.

7.8.4.2 Invariants

- The value must be greater than zero if not null.

OCL for Invariants:

```
inv "positive": isNotNull implies value > 0
```

7.8.5 CO (Coded Ordinal)

7.8.5.1 Description

Specializes QTY

Represents data where coded values are associated with a specific order.

Note: CO may be used for things that model rankings and scores, e.g. likert scales, pain, Apgar values, etc, where there is a) implied ordering, b) no implication that the distance between each value is constant, and c) the total number of values is finite. CO may also be used in the context of an ordered code system. In this case, it may not be appropriate or even possible to use the value attribute, but CO may still be used so that models that make use of such code systems may introduce model elements that involve statements about the order of the terms in a domain.

The relative order of values in a code system need not be independently obvious in the literal representation of the CO. It these circumstances, is expected that an application will look up the ordering of these values from some definition of the code system.

Some of the code systems will directly assign numerical value to the concepts that are suitable for some mathemetical operations.

© ISO 2007 – All rights reserved

Though it would generally make sense, applications SHOULD not assume that the translations of the code, if provided, will have the same ordering as the CO. Translations SHALL not be considered when the ordering of the code system is determined.

7.8.5.2 11404 Syntax

```
type CO = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  value : Real,
  code : CD
)
```

7.8.5.3 Attributes

7.8.5.3.1 value : Real: A numerical value associated with the coded ordinal value.

The value may be constrained to an integer in some contexts of use. If code is nonNull, value SHALL only be nonNull if the code system explicitly assigns a value to the concept.

7.8.5.3.2 code : CD: A code representing the definition of the ordinal item.

7.8.5.4 Equality

Two nonNull CO values are equal if their codes are equal.

Note 1: CO values that have value alone with no code are never equal, as it is not clear whether they are comparable ordinals.

Note 2: since the determination of CO equality is based upon the code, CO values may be equal to CD values.

7.8.5.5 Invariants

- there must be a code or a value if not null

OCL for invariants:

```
inv "must have a code or a value": isNotNull implies
    (code.isNotNull or
     value.oclIsDefined)
inv "No History or Update Mode":noUpdateOrHistory(code)
```

nullFlavor invariants

© ISO 2007 – All rights reserved

```
inv "no value if null": isNull implies
   value.oclIsUndefined
inv "no code if null": isNull implies
   code.oclIsUndefined or code.isNull
```

7.8.5.6 Operations

7.8.5.6.1 max(other : CO) : CO: The maximum of this and other.

Note: If the value attribute is not specified, the applicable terminology may need to be consulted to determine the order of the two values. If no order is defined, the result will be nullFlavor NI.

7.8.5.6.2 min(other : CO) : CO: The minimum of this and other.

Note: If the value attribute is not specified, the applicable terminology may need to be consulted to determine the order of the two values. If no order is defined, the result will be nullFlavor NI.

7.8.5.6.3 compares(other : QTY):BL: This is false unless this and other have the same codeSystem, and then only if the codeSystem defines an order amongst the codes.

7.8.5.6.4 plus[+](other : QTY):QTY: This operation returns null unless this.compares(other) is true, and the codeSystem defines the meaning of addition for these codes.

7.8.5.6.5 minus[-](other : QTY):QTY: This operation returns null unless this.compares(other) is true, and the codeSystem defines the meaning of subtraction for these codes.

7.8.5.7 Examples

```
<example xsi:type="CO" value="1">
  <code code="1" codeSystem="2.16.840.1.113883.2.6.15.1.1">
    <displayName value="Poor"/>
  </code>
</example>
```

In this case, the value "poor“ is assigned a numerical value of 1.

7.8.6 REAL (Real)

7.8.6.1 Description

Specializes QTY

Fractional numbers. Typically used whenever quantities are measured, estimated, or computed from other real numbers. The typical representation is decimal, where the number of significant decimal digits is known as the precision.

7.8.6.2 ISO 11404 Syntax:

```
type REAL = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
```

© ISO 2007 – All rights reserved

```
    controlActExtension : characterstring,
    nullFlavor : NullFlavor,
    updateMode : UpdateMode,
    flavorId : characterstring,
    expression : ED,
    originalText : ED.TEXT,
    uncertainty : QTY,
    uncertaintyType : UncertaintyType,
    value : real,
    precision : integer
  )
```

7.8.6.3 Attributes

7.8.6.3.1 value : Real: The value of the REAL if non Null.

Real is the UML Kernel Real type.

Note: This specification imposes no limitations on the size or precision of REAL, but most implementations will map this to a 8 to 12 byte Real.

7.8.6.3.2 precision : Integer: The number of significant digits of the decimal representation. Null and a value of 0 means nothing is known or stated about the precision.

Notes:

- The precision attribute is only the precision of a decimal digit representation, not the accuracy of the real number value. The purpose of the precision property for the real number data type is to faithfully capture the whole information presented to humans in a number. The amount of decimal digits shown conveys information about the uncertainty (i.e., precision and accuracy) of a measured value.

- The rules for what digits are significant are as follows:

  1. All non-zero digits are significant.

  2. All zeroes to the right of a significant digit are significant.

  3. When all digits in the number are zero the zero-digit immediately left to the decimal point is significant (and because of rule 2, all following zeroes are thus significant too).

  NOTE: These rules of significance differ slightly from the more casual rules taught in school. Notably trailing zeroes before the decimal point are consistently regarded significant here. Elsewhere, e.g., 2000 is ambiguous as to whether the zeroes are significant. This deviation from the common custom is warranted for the purpose of unambiguous communication.

  Examples for the Precision of Real Number Literals.

Literal	Number of Significant Digits
2000	has 4 significant digits.
2e3	has 1 significant digit, used if one would naturally say "2000" but precision is only 1.
0.001	has 1 significant digit.
1e-3	has 1 significant digit, use this if one would naturally say "0.001"

© ISO 2007 – All rights reserved

	but precision is only 1.
0	has 1 significant digit.
0.0	has 2 significant digits.
000.0	has 2 significant digits.
0.00	has 3 significant digits.
4.10	has 3 significant digits.
4.09	has 3 significant digits.
4.1	has 2 significant digits.

- The precision of the representation is independent from uncertainty (precision accuracy) of a measurement result. If the uncertainty of a measurement result is important, one should specify uncertain values as PPD or CIVL.

  The precision of the representation SHOULD match the uncertainty of the value. However, precision of the representation and uncertainty of the value are separate independent concepts. Refer to PPD<REAL> for details about uncertain real numbers.

  For example "0.123" has 3 significant digits in the representation, but the uncertainty of the value may be in any digit shown or not shown, i.e., the uncertainty may be 0.123±0.0005, 0.123±0.005 or 0.123±0.00005, etc. Note that ITS representations should adjust their representational precision with the uncertainty of the value. However, since the precision in the digit string is granular to 0.5 the least significant digit, while uncertainty may be anywhere between these "grid lines", 0.123±0.005 would also be an adequate representation for the value between 0.118 and 0.128.

7.8.6.4 Equality

Two nonNull REAL are equal if they are not null and have the same value and precision.

7.8.6.5 Invariants

- a value must be provided if not null

OCL for Invariants:

```
       inv "null or value": isNull xor value.oclIsDefined

nullFlavor invariants
       inv "no precision if null": isNull implies
           precision.oclIsUndefined
```

© ISO 2007 – All rights reserved

7.8.6.6 Operations

7.8.6.6.1 plus[+] (other : REAL) : REAL: The value of the addition of this and other.

7.8.6.6.2 minus[-] (other : REAL) : REAL: The value of the subtraction of other from this.

7.8.6.6.3 times[*] (other : REAL) : REAL: The value of the multiplication of this and other.

7.8.6.6.4 negated[-] : REAL: The negative value of this.

7.8.6.6.5 dividedBy[/] (other : REAL) : REAL: The value of this divided by other. If other is 0, then the result is nullFlavor NI.

7.8.6.6.6 abs() : REAL: The absolute value of this.

7.8.6.6.7 floor() : INT: The largest integer which is less than or equal to this.

7.8.6.6.8 ceiling() : INT: The smallest integer which is greater than or equal to this.

7.8.6.6.9 round() : INT: The integer which is closest to this. When there are two such integers, the largest one.

7.8.6.6.10 inverted() : REAL: The value of 1 divided by self**.**.

7.8.6.6.11 max(other : REAL) : REAL: The maximum of this and other.

7.8.6.6.12 min(other : REAL) : REAL: The minimum of this and other.

7.8.6.6.13 power(other : REAL) : REAL : this raised to the power of other.

7.8.6.6.14 toInterval() : IVL(REAL) : Converts this value to an interval that expresses the range covered by the precision.

7.8.6.7 Examples

7.8.6.7.1 Precision

```
<example xsi:type="REAL" value="23.0005"/>
```

The floating value 23.0005.

```
<example xsi:type="REAL" value="23" precision="3"/>
```

The floating value 23.000.

7.8.6.7.2 Uncertainty

```
<example xsi:type="REAL" value="23" uncertaintyType="N">
  <uncertainty xsi:type="REAL" value="0.87"/>
</example>
```


© ISO 2007 – All rights reserved

The floating value 23. The uncertainty is known to be a normal distribution with a standard deviation of 0.87. Note that uncertainty will always need an xsi:type.

7.8.7 RTO (Ratio)

Description

Specializes QTY

A quantity constructed as the quotient of a numerator quantity divided by a denominator quantity.

Common factors in the numerator and denominator are not automatically cancelled out.

The RTO datatype supports titers (e.g., "1:128") and other quantities produced by laboratories that truly represent ratios. Ratios are not simply "structured numerics", particularly blood pressure measurements (e.g. "120/60") are not ratios.

Notes:

1. Ratios are different from rational numbers, i.e., in ratios common factors in the numerator and denominator never cancel out. A ratio of two real or integer numbers is not automatically reduced to a real number. This datatype is not defined to generally represent rational numbers. It is used only if common factors in numerator and denominator are not supposed to cancel out. This is only rarely the case. For observation values, ratios occur almost exclusively with titers. In most other cases, REAL should be used instead of the RTO.

2. Since many implementation technologies expect generics to be collections, or only have one parameter, RTO is not implemented as a generic in this specification. Constraints at the point where the RTO is used will define which form of QTY are used.

7.8.7.1 ISO 11404 Syntax:

```
type RTO = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  numerator : QTY,
  denominator : QTY
)
```

© ISO 2007 – All rights reserved

7.8.7.2 Attributes

7.8.7.2.1 numerator : QTY: The quantity that is being divided in the ratio.

7.8.7.2.2 denominator : QTY: The quantity that divides the numerator in the ratio.

The denominator SHALL not be zero.

7.8.7.3 Equality

Two nonNull *RTO*s are equal if their numerator and denominator are equal.

7.8.7.4 Invariants

- If the RTO is not null, both a numerator and a denominator are required
- Uncertainty SHALL not be populated

OCL for Invariants:

```
inv "numerator and denominator required": isNull xor
     (numerator.isNotNull and denominator.isNotNull)
inv “no uncertainty”: uncertainty.isNull
inv "no updateMode or History on RTO Attributes":
      noUpdateOrHistory(numerator) and
       noUpdateOrHistory(denominator)
```

7.8.7.5 Operations

7.8.7.5.1 compares(other : QTY):BL: This and other can be be compared if both the numerator and denominator can be compared.

7.8.7.6 Examples

```
<example xsi:type="RTO">
        <numerator xsi:type="MO" value="103.00" currency="USD"/>
        <denominator xsi:type="PQ" value="1" unit="day"/>
</example>
```

US$103/day.

Note that the inner xsi:type declarations are always required.

7.8.8 PQV (Physical Quantity Value)

7.8.8.1 Description

Abstract. Specializes QTY

An abstract precursor type for PQ and PQR.

© ISO 2007 – All rights reserved

7.8.8.2 ISO 11404 Syntax:

```
type PQ = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  value : real,
  precision : integer,
  codingRationale : Set(CodingRationale),
  source : PQV
)
```

7.8.8.3 Attributes

7.8.8.3.1 value : Real: the number which is multiplied by the unit to make the PQ or PQR value if non Null.

Note This specification imposes no limitations on the size or precision of REAL, but most implementations will map this to a 8 – 12 byte Real.

7.8.8.3.2 precision : Integer: The number of significant digits of the decimal representation. Null and a value of 0 means nothing is known or stated about the precision.

For further information, refer to section 7.8.6.3.2 (REAL.precision).

7.8.8.3.3 codingRationale : Set(CodingRationale): The reason that this PQ or PQR was provided. More than one reason may be given. For possible values, see 7.5.1.4.10 CD.codingRationale.

7.8.8.3.4 source : PQV: A reference to the PQV (PQ or PQR) that was the source of this translation, if this PQV was created by translating it from another. This property is a reference, and if used, must be a reference to a PQV within the same cluster of PQ & PQRs that represent the same physcial quantity. Specifically, the source SHALL be provided within the scope of this physical quantity‘s root and translations.

A PQ consists of a single root quantity and a set of translations, which do not have translations. Using the codingRationale property, a sender can indicate which is the original code. There are some circumstances in which it is useful to know which PQV was translation from which PQV. The source allows for the translation sequence from one translation to another to be represented. Each element of the translation set was translated from the original PQV. Each translation may, however, also have further translations. Thus, when a code is translated multiple times the information about which code served as the input to which translation will be preserved.

7.8.8.4 Equality

No definition of equality is provided for PQV as it is an abstract type.

© ISO 2007 – All rights reserved

7.8.8.5 Invariants

- If the PQV is not null, a value must be present

OCL for Invariants:

```
        inv "null or value": isNull xor value.oclIsDefined
```

nullFlavor invariants

```
        inv "no precision if null": isNull implies
             precision.oclIsUndefined
```

7.8.9 PQ (Physical Quantity)

7.8.9.1 Description

Specializes QTY

A dimensioned quantity expressing the result of measuring.

7.8.9.2 ISO 11404 Syntax:

```
    type PQ = class (
      validTimeLow : characterstring,
      validTimeHigh : characterstring,
      controlActRoot : characterstring,
      controlActExtension : characterstring,
      nullFlavor : NullFlavor,
      updateMode : UpdateMode,
      flavorId : characterstring,
      expression : ED,
      originalText : ED.TEXT,
      uncertainty : QTY,
      uncertaintyType : UncertaintyType,
      value : real,
      precision : integer,
      codingRationale : Set(CodingRationale),
      source : PQV,
      unit : characterstring,
      translation : Set(PQR)
    )
```

7.8.9.3 Attributes

7.8.9.3.1 unit : Code: The unit of measure specified in the Unified Code for Units of Measure (UCUM).
UCUM defines two forms of expression, case sensitive and case insensitive. *PQ* uses the case sensitive codes. The codeSystem OID for the case sensitive form is 2.16.840.1.113883.6.8. The default value for unit is the UCUM code “1” (unity).

Equality of physical quantities does not require the values and units to be equal independently. Value and unit is only how we represent physical quantities. For example, 1 m equals 100 cm. Although the units are different and the values are different, the physical quantities are equal. Therefore one should never expect a particular unit for a physical quantity but instead allow for automated conversion between different comparable units.

© ISO 2007 – All rights reserved

Note:

The unit SHALL come from UCUM, which only specifies unambiguous measurement units. Sometimes it is not clear how some measurements in healthcare map to UCUM codes. The general pattern for a measurement is *value* unit of **Thing**. In this scheme, the PQ represents the *value* and the unit, and the **Thing** is described by some coded concept that is linked to the PQ by the context of use. This maps obviously to some measurements, such as **Patient Body Temperature** of *37* Celsius, and *250* mg/day of **Salicylate**.

However for some measurements that arise in healthcare, the scheme is not so obvious. Two classic examples are 5 Drinks of Beer, and 3 Acetominophen tablets. At first glance it is tempting to classify these measurments like this: *5* drinks of **Beer** and *3* **Acetominophen** tablets. The problem with this is that UCUM does not support units of "beer", "tablets" or "scoops".

The reason for this is that neither tablets or scoops are proper units. What kind of tablets? How big is the glass? In these kinds of cases, the concept that appears to be a unit needs to further specified before interoperability is established. If a correct amount is required, then it is generally appropriate to specify an exact measurement with an appropriate UCUM unit. If this is not possible, then the concept is not part of the measurement. UCUM provides a unit called unity for use in these cases. The proper way to understand these measurements as *3* 1 **Acetominophen** tablets, where 1 is the UCUM unit for unity, and the **Thing** has a qualifier. The context of use will need to provide the extra qualifying information.

7.8.9.3.2 translation : Set(PQR): An alternative representation of the same physical quantity expressed in a different unit from a different unit code system and possibly with a different value.

7.8.9.4 Equality

Two PQ values are equal if the value, precision, and units of their canonical forms are equal.The attributes codingRationale, source, and any translations do not participate in the determination of equality.

7.8.9.5 Invariants

OCL for Invariants:

```
nullFlavor invariants
        inv "no translation if null": isNull implies
             translation->isEmpty
```

7.8.9.6 Operations

7.8.9.6.1 canonical : PQ: The value converted to the form with canonical units. UCUM provides more information about canonical units.

7.8.9.6.2 compares(other : QTY):BL: This and other can be compared if the units of the canonical forms are the same.

© ISO 2007 – All rights reserved

7.8.9.6.3 inverted():PQ: The inverted value of the PQ. Both value and unit must be inverted.

7.8.9.6.4 negated[-] : PQ: The negative value of this.

7.8.9.6.5 plus[+] (other : PQ) : PQ: The value of the addition of this and other; if the units do not match, null (NullFlavor NI).

7.8.9.6.6 minus[-] (other : PQ) : PQ: The value of the subtraction of other from this; if the units do not match, null (NullFlavor NI).

7.8.9.6.7 times[*] (other : PQ) : PQ: The value of the multiplication of this and other with appropriate changes to the units.

7.8.9.6.8 times[*] (other : REAL) : PQ: The value of the multiplication of this and other.

7.8.9.6.9 dividedBy[/] (other : PQ) : PQ: The value of this divided by other with appropriate changes to the units. If other is 0, then the result is nullFlavor NI.

7.8.9.6.10 dividedBy[/] (other : REAL) : PQ: The value of this divided by other. If other is 0, then the result is nullFlavor NI.

7.8.9.6.11 abs() : PQ: The absolute value of this.

7.8.9.6.12 max(other : PQ) : PQ: The maximum of this and other; if the units do not match, null (NullFlavor NI).

7.8.9.6.13 min(other : PQ) : PQ: The minimum of this and other; if the units do not match, null (NullFlavor NI).

7.8.9.6.14 toInterval() : IVL(PQ) : Converts this value to an interval that expresses the range covered by the precision of the value.

7.8.9.7 Examples

7.8.9.7.1 Plain Value

```
<example xsi:type="PQ" value="1.1" unit="mg/mL"/>
```

1.0 mg/mL.

```
<example xsi:type="PQ" value="1.1" unit="mg/mL" codingRationale="R">
  <translation codingRationale="O" value="5">
    <unit codeSystem="2.16.840.1.113883.6.8" code="mM"/>
  </translation>
</example>
```

1.0 mg/mL as a translation from the original measurement of 5mM into UCUM units.

7.8.9.7.2 Using Expressions

```
<substanceAdministration>
   ...
   <doseQuantity xsi:type="PQ" nullFlavor="DER" unit="mL">
     <expression mediatype="application/hl7-factor+xml">
       <xml>
```


© ISO 2007 – All rights reserved

```
        <coefficient value="30" unit="mL/kg"/>
        <factor value="bodyMass"/>
      </xml>
  </doseQuantity>
  ...
  <derivedFrom>
    <localVariableName value="bodyMass"/>
    <monitoringObservation>
      <code code="29463-7" codeSystem="2.16.840.1.113883.11.16492">
         <displayName value="BODY WEIGHT:MASS:PT:^PATIENT:QN"/>
      </code>
    </monitoringObservation>
  </derivedFrom>
</substanceAdministration>
```

This example uses an HL7 specific language to illustrate how the expression attribute is used.

The dose quantity of the substance that is being administered depends on the patient's body mass, 30 mg per kg of body mass. Rather than providing an actual value for the patient's body mass, since it may be unknown or may change, the max dose quantity is given in terms of an expression. Since no value attribute is provided, a nullFlavor must be provided; DER is the appropriate choice when an expression is provided.

The body mass may be found by checking for any observations matching the LOINC code 29463-7; if no matching observations can be found, the outcome of the expression will be null, as the value cannot be known.

Note: Factor is an HL7 specific language documented in the Abstract Datatypes Specification.

7.8.10 PQR (Physical Quantity Representation)

7.8.10.1 Description

Specializes CD

An extension of the coded value datatype representing a physical quantity using a unit from any code system. Used to show alternative representation for a physical quantity. The coded value represents the unit (usually in some other coding system than UCUM).

7.8.10.2 ISO 11404 Syntax:

```
type PQR = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  value : real,
  precision : integer,
  codingRationale : Set(CodingRationale),
  source : PQV,
```

© ISO 2007 – All rights reserved

```
    unit : CD.CV
)
```

7.8.10.3 Attributes

7.8.10.3.1 unit : CD.CV: The alternate unit in which this representation of the physical quantity is encoded.
The unit SHALL not have translations.

7.8.10.4 Equality

Two PQR values are equal if their value, precision and unit are all equal. The attributes codingRationale and source do not participate in the determination of equality.

7.8.10.5 Invariants

- A unit is required.
- No originalText.

  Note: There is only one originalText, that for the physical quantity itself. The code carries it's own originalText which is the originalText for the unit, not the whole PQ

OCL for Invariants:

```
inv "null or unit": isNull xor unit.isNotNull

inv "no originalText": originalText.oclIsUndefined
```

nullFlavor invariants

```
inv "no updateMode or History on PQR": noUpdateOrHistory
```

7.8.11 PQ.TIME

7.8.11.1 Description

Constrains PQ

PQ.TIME constraints PQ so that it must have units that describe a period of time.

7.8.11.2 Invariants

- The units must be a measure of time ("such as, "s" (second), "min", "h" (hour), "d" (day), "wk" (week), "a" (year))

OCL for Invariants:

© ISO 2007 – All rights reserved

7.8.12 MO (Monetary Amount)

7.8.12.1 Description

Specializes QTY

A MO is a quantity expressing the amount of money in some currency.

Currencies are the units in which monetary amounts are denominated in different economic regions. While the monetary amount is a single kind of quantity (money) the exchange rates between the different units are variable. This is the principle difference between PQ and MO, and the reason why currency units are not physical units.

7.8.12.2 ISO 11404 Syntax:

```
type MO = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  value : real,
  precision : integer,
  currency : characterstring
)
```

7.8.12.3 Attributes

7.8.12.3.1 value : Real The value of the MO if non Null. MO values are usually precise to 0.01 (one cent, penny, paisa, etc.) For large amounts, it is important not to store MO values in floating point registers, since this may lose precision.

7.8.12.3.2 precision : Integer: The number of significant digits of the decimal representation. null and a value of 0 means nothing is known or stated about the precision.

For further information, refer to section 7.8.6.3.2 (REAL.precision).

7.8.12.3.3 currency : Code: The currency unit as defined in ISO 4217.

7.8.12.4 Equality

Two *MO* values are equal if their value, precision and currency attributes are equal.

7.8.12.5 Invariants

- If not null, a value must be present

© ISO 2007 – All rights reserved

- If not null, a currency must be defined

OCL for Invariants:

```
        inv "null or value": isNull xor value.oclIsDefined
        inv "null or currency": isNull xor currency.oclIsDefined
```

nullFlavor invariants

```
        inv "no precision if null": isNull implies
             precision.oclIsUndefined
```

7.8.12.6 Operations

7.8.12.6.1 plus[+] (other : MO) : MO: The value of the addition of this and other; if the currencies do not match, null (NullFlavor NI).

7.8.12.6.2 minus[-] (other : MO) : MO: The value of the subtraction of other from this; if the currencies do not match, null (NullFlavor NI).

7.8.12.6.3 times[*] (other : REAL) : MO: The value of the multiplication of this by other.

7.8.12.6.4 dividedBy[/] (other : REAL) : MO: The value of this divided by other. If other is 0, then the result is nullFlavor NI.

7.8.12.6.5 max(other : MO) : MO: The maximum of this and other.

7.8.12.6.6 min(other : MO) : MO: The minimum of this and other.

7.8.12.6.7 compares(other : QTY):BL: This and other can be compared if their currencies are the same.

7.8.12.7 Examples

```
<example xsi:type="MO" value="42" currency="AUD"/>
```

$42 Australian dollars.

7.8.13 TS (Point in Time)

7.8.13.1 Description

Specializes QTY

A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression.

7.8.13.2 ISO 11404 Syntax:

```
    type TS = class (
      validTimeLow : characterstring,
      validTimeHigh : characterstring,
      controlActRoot : characterstring,
```


© ISO 2007 – All rights reserved

```
    controlActExtension : characterstring,
    nullFlavor : NullFlavor,
    updateMode : UpdateMode,
    flavorId : characterstring,
    expression : ED,
    originalText : ED.TEXT,
    uncertainty : QTY,
    uncertaintyType : UncertaintyType,
    value : characterstring
  )
```

7.8.13.3 Attributes

7.8.13.3.1 value : String: The value of the TS if non Null. value is a string with the format "YYYYMMDDHHMMSS.UUUU[+|-ZZzz]" that conforms to the constrained ISO 8601 that is defined in ISO 8824 (ASN.1) under clause 32 (generalized time). The format should be used to the degree of precision that is appropriate.

7.8.13.4 Equality

Two nonNull TS values are only equal if their time and precision are equal. If both TS value have timezones, the values should be be corrected for timezone before comparison. If neither TS value has a timezone, then they may be compared for equality without correction. If only one TS value has a timezone, then the equality is null (nullFlavor NI).

7.8.13.5 Invariants

- If the TS is not null, a value must be present

OCL for Invariants:

```
def: let hasTimezone : Boolean = value.pos("+") > 0
    or value.pos("-") > 0
inv "null or value": isNull xor value.oclIsDefined
```

7.8.13.5.1 Operations

7.8.13.5.2 plus(+) (other : PQ) : TS: The value of the addition of this and other; if other.units are not a time, null (NullFlavor NI).

7.8.13.5.3 minus[-] (other : PQ) : TS: The value of the subtraction of other from this; if other.units are not a time, null (NullFlavor NI).

7.8.13.5.4 minus[-] (other : TS) : PQ: The value of the subtraction of other from this; the return value will have units that are a time.

7.8.13.5.5 max(other : TS) : TS: The maximum of this and other.

7.8.13.5.6 min(other : TS) : TS: The minimum of this and other.

7.8.13.5.7 toInterval() : IVL(TS) : Converts this value to an interval that expresses the range covered by the precision.

© ISO 2007 – All rights reserved

7.8.13.6 Examples

7.8.13.6.1 Instant in Time

```
<example xsi:type="TS" value="20031101234511+0500"/>
```

11:45pm on 01-Nov 2003 at +5 from UTC (e.g. US eastern).

7.8.13.6.2 Birth Date

```
<example xsi:type="TS" flavorId="TS.BIRTH" value="1945"/>
```

Patient was born in 1945. Month and day are unknown.

The outcome of toIVL() for this example would be the following interval:

```
<example xsi:type="IVL_TS" lowClosed="true" highClosed="false">
  <low value="194501010000.0000"/>
  <high value="19460101000.0000"/>
</example>
```

7.8.14 TS.DATE

7.8.14.1 Description

Constrains TS

TS.DATE constrains TS so that it may only contain a date value.

7.8.14.2 Invariants:

- No Timezone
- No hours, minutes, seconds or milliseconds

OCL for Invariants

```
inv "Date": not hasTimezone and value.size <= 8
```

7.8.15 TS.DATE.FULL

7.8.15.1 Description

Constrains TS.DATE

TS.DATE.FULL constrains TS.DATE so that it must contain reference to a particular day.

7.8.15.2 Invariants:

- A full day must be specified

OCL for Invariants

© ISO 2007 – All rights reserved

```
inv "Full Date": value.size = 8
```

7.8.16 TS.DATETIME

7.8.16.1 Description

Constrains TS

TS.DATETIME constrains a TS so that it's precision can not be more precise than seconds.

7.8.16.2 Invariants:

- Milliseconds must be blank

OCL for Invariants

```
inv "DateTime": (value.size <= 14) or (hasTimezone and
     value.size <= 19)
```

7.8.17 TS.DATETIME.FULL

7.8.17.1 Description

Constrains TS.DATETIME

TS.DATETIME.FULL constrains TS.DATETIME so that it must contain reference to a particular second with a timezone.

7.8.17.2 Invariants

- A timezone is required
- A full time, including seconds and timezone, is required

OCL for Invariants

```
inv "Full DateTime": value.size = 19 and hasTimezone
```

7.8.18 TS.BIRTH

Description

Constrains TS

TS.BIRTH constrains TS so that it must refer to a birth date.

Note: The generally accepted convention for birth dates is that either a year, a particular day, or a fully specified date time may be known.

© ISO 2007 – All rights reserved

7.8.18.1 Invariants

- Either a full day or a full time with a timezone is required

OCL for Invariants

```
inv "Birth": (hasTimezone and value.size = 19) or
      (not hasTimezone and (value.size = 8 or value.size = 4))
```


© ISO 2007 – All rights reserved

7.10 Collection Datatypes

These data types provide support for collections of data.

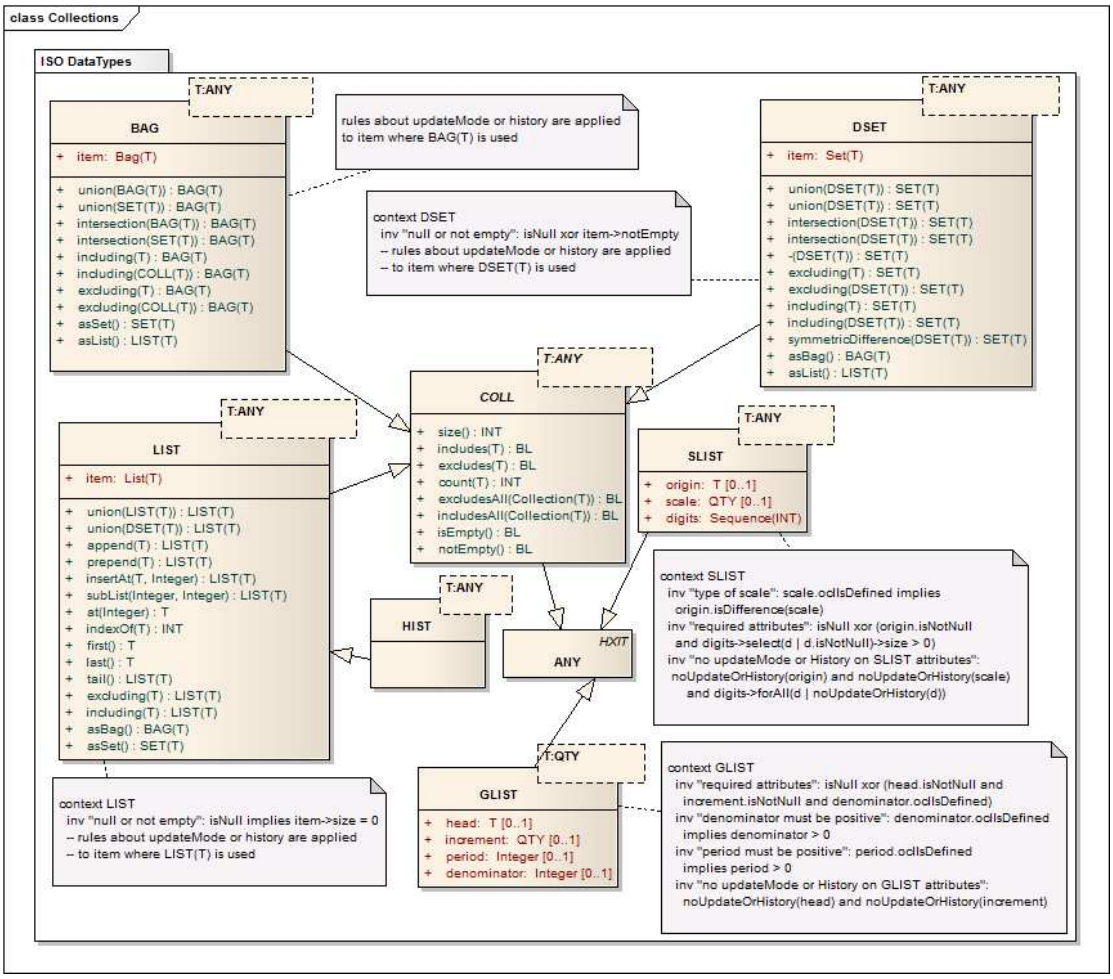


Figure 10. Collection data types

7.10.1 COLL

7.10.1.1 Description

Abstract. Specializes ANY.

Parameter: T : ANY

This is an abstract type introduced to gather common properties and behavior of the collection classes.

© ISO 2007 – All rights reserved

7.10.1.2 Operations

7.10.1.2.1 size() : INT: The number of elements in this collection.

7.10.1.2.2 includes(object : T) : BL: True if object is an element of this collection.

7.10.1.2.3 excludes(object : T) : BL: True if object is not an element of this collection.

7.10.1.2.4 count(object : T) : INT: The number of times that object occurs in the this collection.

7.10.1.2.5 includesAll(c2 : Collection(T)) : BL: True if this collection contains all the elements of c2.

7.10.1.2.6 excludesAll(c2 : Collection(T)) : BL: True if this collection contains none of the elements of c2.

7.10.1.2.7 isEmpty() : BL: True if this collection is the empty collection.

7.10.1.2.8 notEmpty() : BL: True if this collection is not the empty collection.

7.10.1.3 Equality

Equality is not defined for COLL as it is an abstract type.

7.10.2 DSET (Discrete Set)

7.10.2.1 Descripion

Specializes COLL

Parameter: T : ANY

A collection that contains other distinct and discrete values in no particular order.

Sets SHALL not contain duplicate items. The context of use SHALL define how elements are compared when checking set element uniqueness.

In most cases, sets will not contain null values as it cannot be determined whether they are duplicates or not; however this depends on the definition of uniqueness for the set elements.

7.10.2.2 ISO 11404 Syntax:

```
type DSET (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  item : Set(T)
)
```


© ISO 2007 – All rights reserved

Note: DSET specifies a set of discrete items. If the items are in an ordered but discrete domain (i.e. INT) and they represent a sequence of concurrent values such as 2,3,4,5, then the same set can also be specified by a QSET such as IVL(2..5). However for all other cases, there is no overlap between QSET and DSET.

7.10.2.3 Attributes

7.10.2.3.1 item : Set(T): The contents of the Set.

7.10.2.4 Equality

Two nonNull DSETs are equal if they contain the same elements.

Note 1: the determination of element content is based on the same semantic equals as defined in this specification, so it is possible that a SET(CD) can be equal to SET(CS), for instance.

Note 2: It is possible for a DSET<INT> and a IVL<INT> to be equal, if they both contain the same elements.

7.10.2.5 Invariants

- If the DSET is null, no items can be contained in the Set. If the DSET is not null, at least one item must be present

OCL for Invariants:

```
inv "null or not empty": isNull xor item->notEmpty
```

© ISO 2007 – All rights reserved

7.10.2.6 Operations

7.10.2.6.1 union(s : DSET(T)) : DSET(T): The union of this and s.

7.10.2.6.2 union(bag : BAG(T)) : BAG(T): The union of this and bag. Duplicate items in bag will be eliminated.

7.10.2.6.3 intersection(s : DSET(T)) : DSET(T): The intersection of this and s (i.e, the SET of all elements that are in both this and s).

7.10.2.6.4 intersection(bag : BAG(T)) : DSET(T): The intersection of this and bag.

7.10.2.6.5 minus[–](s : DSET(T)) : DSET(T): The elements of this, which are not in s.

7.10.2.6.6 including(object : T) : DSET(T): The SET containing all elements of this plus object if it is not already in the set.

7.10.2.6.7 including(other : DSET(T)) : DSET(T): The SET containing all elements of this plus all the elements of other, with any duplicates removed .

7.10.2.6.8 excluding(object : T) : DSET(T): The SET containing all elements of this without object.

7.10.2.6.9 excluding(other : DSET(T)) : DSET(T): The SET containing all elements of this with any elements of other removed .

7.10.2.6.10 symmetricDifference(s : DSET(T)) : DSET(T): The sets containing all the elements that are in this or s.

7.10.2.6.11 asList() : LIST(T): A Sequence that contains all the elements from this, in undefined order.

7.10.2.6.12 asBag() : BAG(T): The Bag that contains all the elements from this.

7.10.2.7 Examples

7.10.2.7.1 Integer Sets

```
<example xsi:type="DSET_INT">
  <item value="3"/>
  <item value="6"/>
  <item value="9"/>
  <item value="11"/>
</example>
```

The set of integers (3,6,9,11). Note that the set type specifies the type of the items in the set (xsi:type="DSET" is not correct as SET is a generic type).

```
<example xsi:type="DSET_INT">
  <item value="11"/>
  <item value="6"/>
  <item value="9"/>
  <item value="3"/>
</example>
```


© ISO 2007 – All rights reserved

This set is identical to the previous set - order has no significant in sets.

7.10.2.7.2 Problems with sets

```
<example xsi:type="DSET_TEL">
  <item value="tel:+15556667777" use="H"/>
  <item nullFlavor="UNK" use="WP"/>
</example>
```

A set of telephone numbers, with a known home phone number, and an unknown work number. IMPORTANT: This is an illegal set: Sets cannot contain nullValues. In general, addresses and telephone numbers should not be modeled as DSETs for this reason.

```
<example xsi:type="DSET_REAL">
  <item value="3.1"/>
  <item value="3.5"/>
</example>
```

Where the items are not discrete (PQ, MO, REAL), a discrete set is rather ambiguous. For example, is 3.103 a member of this set or not? (semantically it should be, but by the definitions contained in the abstract datatypes, it isn't). For this reason, DSET should not be used with these types.

7.10.3 LIST (Sequence)

7.10.3.1 Description

Specializes COLL

Parameter: T : ANY

A collection that contains other discrete (but not necessarily distinct) values in a defined sequence.

The sequence is an ordered collection of values, but no particular order is inherently associated with the sequence. The meaning of the order of the items SHOULD be defined where a LIST is used. Note that in some cases, the order is fixed (e.g. HIST), but in other cases, the order is not fixed: only the meaning associated with the order in the instance is defined (e.g. EN, AD).

7.10.3.2 ISO 11404 Syntax:

```
type LIST (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
```

© ISO 2007 – All rights reserved

```
    item : Sequence(T)
  )
```

7.10.3.3 Attributes

7.10.3.3.1 item : Sequence(T): The contents of the Sequence.

7.10.3.4 Equality

Two lists are equal if and only if they are both empty, or if they contain the same elements in the same order.

Note 1: the determination of element content is based on the same semantic equals as defined in this specification, so it is possible that a LIST(ED) can be equal to LIST(ST), for instance.

Note 2: SLIST equality is based on the equality of a sequence of values, so a SLIST<T> may be equal to a LIST<T>.

7.10.3.5 Invariants

OCL for Invariants:

```
inv "null or not empty": isNull implies item->size = 0
```


© ISO 2007 – All rights reserved

7.10.3.6 Operations

7.10.3.6.1 union (s : LIST(T)) : LIST(T): The LIST consisting of all elements in this, followed by all elements in s.

7.10.3.6.2 union (s : DSET(T)) : LIST(T): The LIST consisting of all elements in this, followed by all elements in s in some arbitrary order.

7.10.3.6.3 append (object: T) : LIST(T): The LIST of elements, consisting of all elements of this, followed by object.

7.10.3.6.4 prepend(object : T) : LIST(T): The LIST consisting of object, followed by all elements in this.

7.10.3.6.5 insertAt(object : T, index : Integer) : LIST(T): The LIST consisting of this with object inserted at position index.

7.10.3.6.6 subList(lower : Integer, upper : Integer) : LIST(T): The sub-LIST of this starting at number lower, up to and including element number upper.

7.10.3.6.7 at(i : Integer) : T: The i-th element of LIST.

7.10.3.6.8 indexOf(obj : T) : INT: The index of object obj in the LIST.A NullFlavor if the item exists more than once.

7.10.3.6.9 first() : T: The first element in this.

7.10.3.6.10 last() : T: The last element in this.

7.10.3.6.11 tail() : LIST(T): The list with the first element removed.

7.10.3.6.12 including(object : T) : LIST(T): The LIST containing all elements of this plus object added as the last element.

7.10.3.6.13 excluding(object : T) : LIST(T): The LIST containing all elements of this apart from all occurrences of object. The order of the remaining elements is not changed.

7.10.3.6.14 asBag() : BAG(T): The Bag containing all the elements from this, including duplicates, or a nullFlavor as appropriate.

7.10.3.6.15 asSet() : DSET(T): The Set containing all the elements from this, with duplicated removed.

7.10.3.7 Examples

```
<example xsi:type="LIST_INT">
  <item value="3"/>
  <item value="11"/>
  <item value="6"/>
  <item value="9"/>
</example>
```

A list of integers. The order is significant and must always be maintained.

© ISO 2007 – All rights reserved

7.10.4 GLIST (Generated Sequence)

Description

Specializes ANY

Parameter: T : QTY

A periodic or monotone sequence of values generated from a few parameters, rather than being enumerated. Used to specify regular sampling points for biosignals.

7.10.4.1 ISO 11404 Syntax:

```
type GLIST (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  head: T,
  increment : QTY,
  denominator : integer,
  period : integer
)
```

7.10.4.2 Attributes

7.10.4.2.1 head : T: The first item in this sequence. This is the start-value of the generated list.

7.10.4.2.2 increment : QTY: The difference between one value and its previous different value. For

Example: To generate the sequence (1; 4; 7; 10; 13; ...) the increment is 3; likewise to generate the sequence (1; 1; 4; 4; 7; 7; 10; 10; 13; 13; ...) the increment is also 3. The actual type QTY will be dictated by the type of T

7.10.4.2.3 denominator : Integer: The integer by which the index for the sequence is divided, effectively the number of times the sequence generates the same sequence item value before incrementing to the next sequence item value.

Example: To generate the sequence (1; 1; 1; 2; 2; 2; 3; 3; 3; ...) the denominator is 3.

The use of the denominator is to allow multiple generated sequences to periodically scan a multidimensional space. For example, an (abstract) TV screen uses 2 such generators for the columns and rows of pixels. For instance, if there are 200 scan lines and 320 raster colunmns, the column-generator would have denominator 1 and the line-generator would have denominator 320.

© ISO 2007 – All rights reserved

7.10.4.2.4 period : Integer: If non-NULL, specifies that the sequence alternates, i.e., after this many increments, the sequence item values roll over to start from the initial sequence item value.

Example: The sequence (1; 2; 3; 1; 2; 3; 1; 2; 3; ...) has period 3; also the sequence (1; 1; 2; 2; 3; 3; 1; 1; 2; 2; 3; 3; ...) has period 3 too.

The period allows to repeatedly sample the same sample space. The "waveform" of this periodic generator is always a "saw", just like the x-function of your oscilloscope.

7.10.4.3 Equality

GLIST is a list generator. Two GLISTs are equal if they specify the same sequence of elements.

7.10.4.4 Invariants

- If the GLIST is not null, all attributes but the period are required
- Denominator must be positive
- The period must be positive

OCL for Invariants:

```
inv "required attributes": isNull xor (head.isNotNull and
     increment.isNotNull and denominator.oclIsDefined)
inv "denominator must be positive": denominator.oclIsDefined
      implies denominator > 0
inv "period must be positive": period.oclIsDefined implies
      period > 0
inv "no updateMode or History on GLIST attributes":
     noUpdateOrHistory(head) and noUpdateOrHistory(increment)
```

7.10.4.5 Examples

```
<example xsi:type="GLIST_PQ" period="100" denominator="100">
  <head value="0" unit="V"/>
  <increment xsi:type="PQ" value="1" unit="mV"/>
</example>
```

The x-wave of a digital oscillograph scanning between 0 and 100 mV in 100 steps of 1 mV. The frequency is unknown from these data as we do not know how much time elapses between each step of the index.

Example 65.

```
<example xsi:type="GLIST_TS" denominator="1">
  <head value="2002072920300"/>
  <increment xsi:type="PQ" value="100" unit="us"/>
</example>
```

A timebase from June 29, 2002 at 8:30 PM with 100 us between each steps of the index. If combined with the previous generator as a second sampling dimension this would now describe our digital oscilloscope's x-timebase as 1 mV per 100 us. At 100 steps per period, the period is 10 ms, which is equal to a frequency of 100 Hz.

© ISO 2007 – All rights reserved

Other examples:

head	incre-ment	deno-minator	period	meaning
0	1	1	∞	The identity-sequence where each item is equal to its index.
198706052000	2 hour	1	∞	Sequence starting on June 5, 1987 at 7 PM and incrementing every two hours: 9 PM, 11 PM, 1 AM (June 6), 3 AM, 5 AM, and so on.
0 V	1 mV	1	100	The x-wave of a digital oscillograph scanning between 0 and 100 mV in 100 steps of 1 mV. The frequency is unknown from these data as we do not know how much time elapses between each step of the index.
2002072920300	100 us	1	∞	A timebase from June 29, 2002 at 8:30 PM with 100 us between each steps of the index. If combined with the previous generator as a second sampling dimension this would now describe our digital oscilloscope's x-timebase as 1 mV per 100 us. At 100 steps per period, the period is 10 ms, which is equal to a frequency of 100 Hz.
0 V	1 mV	100	100	Combining this generator to the previous two generators could describe a three-dimensional sampling space with two voltages and time. This generator also steps at 1 mV and has 100 steps per period, however, it only steps every 100 index increments, so, the first voltage generator makes one full cycle before this generator is incremented. One can think of the two voltages as "rows" and "columns" of a "sampling frame". With the previous generator as the timebase, this results in a scan of sampling frames of 100 mV × 100 mV with a framerate of 1 Hz.

7.10.5 SLIST (Sampled Sequence)

7.10.5.1 Description

Specializes ANY

Parameter: T : QTY

A sequence of sampled values scaled and translated from a list of integer values. Used to specify sampled biosignals.

© ISO 2007 – All rights reserved

7.10.5.2 ISO 11404 Syntax:

```
type SLIST (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  origin : T,
  scale : QTY,
  digits : Sequence(INT)
)
```

7.10.5.3 Attributes

7.10.5.3.1 origin : T: The origin of the list item value scale. The physical quantity that a zero-digit in the sequence would represent.

7.10.5.3.2 scale : QTY: A ratio-scale quantity that is factored out of the digit sequence. The actual type QTY will be dictated by the type of T.

7.10.5.3.3 digits : Sequence(INT): A sequence of raw digits for the sample values. This is typically the raw output of an A/D converter.

7.10.5.4 Equality

SLIST specifies a LIST. Two SLISTs are equal if they specify the same sequence of values.

Note: Because SLIST specifies a LIST, and equality is basd on the LIST that is specified, it is possible that a LIST<PQ> will be equal to a SLIST<PQ>.

7.10.5.5 Invariants

- scale must be a difference from the origin
- If the SLIST is not null, an origin and at least one digit is required

OCL for Invariants:

```
inv "required attributes": isNull xor (origin.isNotNull and
     digits->select(d | d.isNotNull)->size > 0)
inv "type of scale": scale.oclIsDefined implies
     origin.isDifference(scale)
inv "no updateMode or History on SLIST attributes":
    noUpdateOrHistory(origin) and noUpdateOrHistory(scale)
     and digits->forAll(d | noUpdateOrHistory(d))
```

7.10.5.6 Examples

```
<example xsi:type="SLIST_PQ">
   <origin value='0' unit='uV'/>
```

© ISO 2007 – All rights reserved

```
<scale xsi:type="PQ" value='2.5' unit='uV'/>
<digit value="-4"/>
<digit value="-13"/>
<digit value="-18"/>
<digit value="-18"/>
<digit value="-18"/>
<digit value="-17"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-17"/>
<digit value="-18"/>
<digit value="-18"/>
<digit value="-1"/>
<digit value="-17"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-16"/>
<digit value="-15"/>
<digit value="-13"/>
<digit value="-11"/>
<digit value="-10"/>
<digit value="-10"/>
<digit value="-9"/>
<digit value="-6"/>
<digit value="-4"/>
<digit value="-5"/>
<digit value="-5"/>
<digit value="-3"/>
<digit value="-2"/>
<digit value="-2"/>
<digit value="-1"/>
<digit value="1"/>
<digit value="2"/>
<digit value="3"/>
<digit value=""/>
<digit value="7"/>
<digit value="8"/>
<digit value="9"/>
<digit value="10"/>
<digit value="11"/>
<digit value="12"/>
<digit value="13"/>
<digit value="15"/>
<digit value="17"/>
<digit value="19"/>
<digit value="21"/>
<digit value="23"/>
<digit value="25"/>
<digit value="27"/>
<digit value="29"/>
<digit value="30"/>
<digit value="30"/>
<digit value="31"/>
<digit value="34"/>
<digit value="37"/>
<digit value="40"/>
<digit value="43"/>
<digit value="45"/>
<digit value="4"/>
<digit value="46"/>
<digit value="46"/>
<digit value="46"/>
<digit value="46"/>
```


© ISO 2007 – All rights reserved

```
<digit value="47"/>
<digit value="49"/>
<digit value="51"/>
<digit value="53"/>
<digit value="55"/>
<digit value="57"/>
<digit value="59"/>
<digit value="60"/>
<digit value="59"/>
<digit value="58"/>
<digit value="58"/>
<digit value="58"/>
<digit value="57"/>
<digit value="56"/>
<digit value="56"/>
<digit value="56"/>
<digit value="57"/>
<digit value="57"/>
<digit value="5"/>
<digit value="53"/>
<digit value="50"/>
<digit value="47"/>
<digit value="45"/>
<digit value="74"/>
<digit value="51"/>
<digit value="38"/>
<digit value="33"/>
<digit value="31"/>
<digit value="28"/>
<digit value="25"/>
<digit value="21"/>
<digit value="16"/>
<digit value="14"/>
<digit value="15"/>
<digit value="13"/>
<digit value="9"/>
<digit value="7"/>
<digit value="4"/>
<digit value="1"/>
<digit value="-1"/>
<digit value="-3"/>
<digit value="-4"/>
<digit value="-6"/>
<digit value="-10"/>
<digit value="-12"/>
<digit value="-13"/>
<digit value="-12"/>
<digit value="-12"/>
<digit value="-17"/>
<digit value="-18"/>
<digit value="-18"/>
<digit value="-18"/>
<digit value="-19"/>
<digit value="-20"/>
<digit value="-21"/>
<digit value="-20"/>
<digit value="-20"/>
<digit value="-20"/>
<digit value="-20"/>
<digit value="-2"/>
...
<digit value="2"/>
<digit value="1"/>
<digit value="0"/>
<digit value="0"/>
<digit value="0"/>
```

© ISO 2007 – All rights reserved

```
   <digit value="1"/>
   <digit value="2"/>
   <digit value="2"/>
   <digit value="1"/>
   <digit value="1"/>
   <digit value="1"/>
   <digit value="0"/>
   <digit value="-1"/>
   <digit value="0"/>
   <digit value="1"/>
   <digit value="1"/>
   <digit value="1"/>
   <digit value="1"/>
   <digit value="2"/>
   <digit nullFlavor="UNK"/>
</example>
```

This example shows Lead II of an EKG tracing, with origin calibrated at 0 µV and with a scale factor of 2.5 µV. The last measurement failed (to show example of nullFlavor).

7.10.6 HIST (History)

7.10.6.1 Description

Specializes LIST

Parameter: T : ANY

A collection that set of items in historical order.

7.10.6.2 11404 Syntax:

```
type HIST (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  item : Set(T)
)
```

Note: The historical information pertains to the correctness of the information rather than the systems knowledge of the actual information. For further information see Section 7.3.2 (HXIT).

All items in the list SHALL have a non-null validTime. The validTime periods on the list SHALL not overlap. The contents of HIST SHALL be ordered in ascending chronological order..

7.10.6.3 Examples

```
<example xsi:type="HIST_TEL">
  <item nullFlavor="UNK" use="W" use="H" validTimeHigh="199206"/>
  <item value="tel:+15552225543" use="H" validTimeLow="199206"
     validTimeHigh="199207"/>
```


© ISO 2007 – All rights reserved

```
  <item value="tel:+15556667777" use="H" validTimeLow="199207"/>
</example>
```

This specifies a known history of home phone numbers.

7.10.7 BAG (Bag)

7.10.7.1 Description

Specializes COLL

Parameter: T : ANY

An unordered collection of values, where each value can be contained more than once in the collection.

7.10.7.2 ISO 11404 Syntax:

```
type BAG (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  item : Bag(T)
)
```

7.10.7.3 Attributes

7.10.7.3.1 item : Bag(T): The contents of the Bag.

7.10.7.4 Equality

Two bags are equal if and only if they are both empty, or if they both contain the same items, with the same number of each item present.

Note: the determination of element content is based on the same semantic equals as defined in this specification, so it is possible that a BAG(CD) can be equal to BAG(CO), for instance.

7.10.7.5 Invariants

Note: Rules about updateMode or history are applied to item where BAG(T) is used

© ISO 2007 – All rights reserved

7.10.7.6 Operations

7.10.7.6.1 union(bag : BAG(T)) : BAG(T): The union of this and bag.

7.10.7.6.2 union(set : DSET(T)) : BAG(T): The union of this and set.

7.10.7.6.3 intersection(bag : BAG(T)) : BAG(T): The intersection of this and bag.

7.10.7.6.4 intersection(set : DSET(T)) : DSET(T): The intersection of this and set.

7.10.7.6.5 including(object : T) : BAG(T): The bag containing all elements of this plus object.

7.10.7.6.6 including(coll : COLL(T)) : BAG(T): The bag containing all elements of this plus any elements in coll.

7.10.7.6.7 excluding(object : T) : BAG(T): The bag containing all elements of this apart from all occurrences of object.

7.10.7.6.8 excluding(coll : COLL(T)) : BAG(T): The bag containing all elements of this apart from any objects that are found in coll.

7.10.7.6.9 asList() : LIST(T): A Sequence that contains all the elements from this, in undefined order.

7.10.7.6.10 asSet() : DSET(T): The Set containing all the elements from this, with duplicates removed.

7.10.7.7 Examples

```
<example xsi:type="BAG_TEL">
  <item value="tel:+15556667777" use="H"/>
  <item nullFlavor="UNK" use="W"/>
</example>
```

A bag of telephone numbers, with a known home phone number, and an unknown work number.

```
<example xsi:type="BAG_TEL">
  <item nullFlavor="UNK" use="W"/>
  <item value="tel:+15556667777" use="H"/>
</example>
```

This is not equal to the previous bag of telephone numbers - although order is not important, the work phone number has a nullFlavor, and equality cannot be evaluated.

© ISO 2007 – All rights reserved

7.11 Continuous Set Data types

These data types provide support for collections of data.

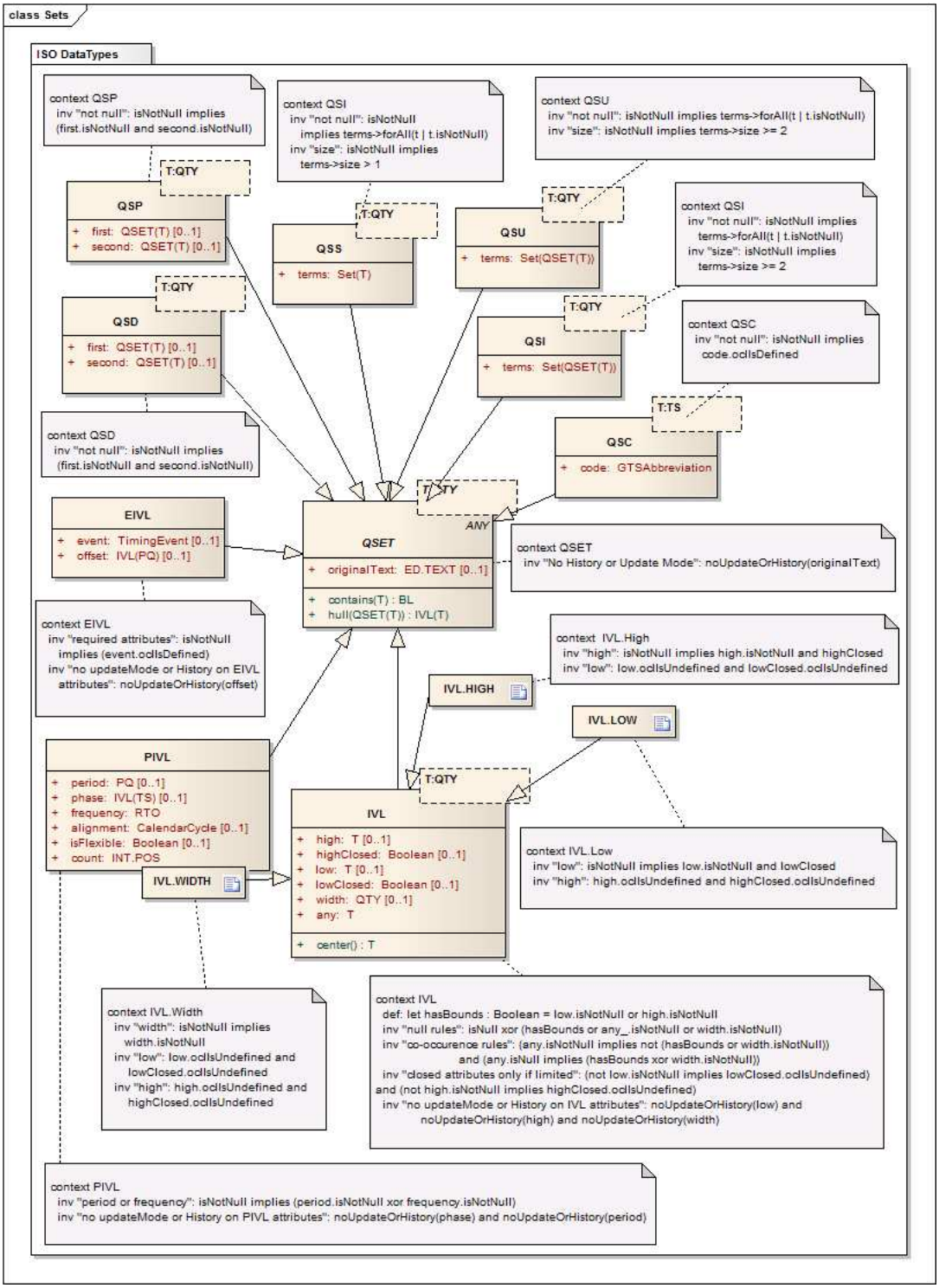


Figure 11. Continuous set data types

© ISO 2007 – All rights reserved

7.11.1 QSET (Continuous Set)

7.11.1.1 Description

Abstract; specializes ANY

Parameter: T : QTY

An unordered set of distinct values which are quantities.

Any ordered type can be the basis of an QSET; it does not matter whether the base type is discrete or continuous. If the base data type is only partially ordered, all elements of the QSET must be elements of a totally ordered subset of the partially ordered data type (for example, PQ is only ordered when the units are consistent. Every value in a QSET(PQ) must have the same canonical unit).

QSET is an abstract type. A working QSET is specified as an expression tree built using a combination of operator (QSI, QSD, QSU, QSP) and component types (QSC, QSS and IVL; and, for TS, PIVL and EIVL).

QSETs SHALL not contain null values as members of the set.

7.11.1.2 ISO 11404 Syntax:

```
    type QSET (T : ANY) = class (
      validTimeLow : characterstring,
      validTimeHigh : characterstring,
      controlActRoot : characterstring,
      controlActExtension : characterstring,
      nullFlavor : NullFlavor,
      updateMode : UpdateMode,
      flavorId : characterstring,
      originalText : ED.TEXT
   )
```

7.11.1.3 Attributes

7.11.1.3.1 originalText : ED.TEXT: The text representation from which the QSET was encoded, if such a representation is the source of the QSET.

Original text can be used in a structured user interface to capture what the user saw as a representation of the set on the data input screen, or in a situation where the user dictates or directly enters text, it is the text entered or uttered by the user.

It is valid to use a QSET derived data type to store only the text that the user entered or uttered. In this situation, original text will exist without a valid value. The originalText is not a substitute for a valid value. If the actual content of the *QSET* is not valid, then the *QSET* SHALL be null, irrespective of whether originalText has a value or not.

Notes

- The details of the link in the originalText.reference between different artifacts of medical information (e.g., document and coded result) is outside the scope of this specification and may be further proscribed in specifications that use this specification.

© ISO 2007 – All rights reserved

- The original text SHALL be an excerpt of the relevant information in the original sources, rather than a pointer or exact reproduction. Thus the original text SHALL be represented in plain text form. In specific circumstances, when clearly descirbed the context of use, the originalText may be a reference to some other text artefact for which the resolution scope is clearly described.

7.11.1.4 Equality

The notional equality determination for QSET and all it's descendents except is based on set membership: two QSETs are equal if they contain the same members. However, QSETs are used to build expression trees that may become quite complex. It is not feasible to determine whether two different QSET expression trees describe the same set of elements, so the determination for whether two QSETs are equal is the default equality test defined in ANY.

This equality test applies to all QSET specializations except for IVL, and is not specified for the other specializations.

7.11.1.5 Invariants

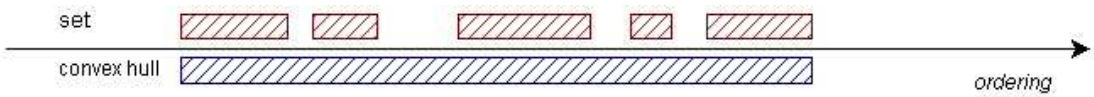
OCL for invariants:

```
inv "No History or Update Mode":
  noUpdateOrHistory(originalText)
```

7.11.1.6 Operations

7.11.1.6.1 contains (x: T): BL: True if the QSET contains the value x.

7.11.1.6.2 hull (x: QSET(T)): IVL(T) : The convex hull of this set with x, which is the smallest interval that is a superset of this and x.



Note: The convex hull of a *QSET* can less formally be called the "outer bound interval". Thus the convex hull of a *QSET* describes the absolute beginning and end of a schedule. For some set specifications, where there is infinite repetitions (e.g., a PIVL(TS)) the convex hull has infinite bounds. The term schedule is used throughout this section in its general sense, that of an organised series of values. The more common meaning of the term schedule, that of a time-based plan of events, is exactly a *QSET*(TS).

7.11.1.7 Examples

7.11.1.7.1 QSET(TS)

The type QSET(TS) is also known as GTS: General Timing Specification.

The first example specifies every other Tuesday in the season from the (US holidays) Memorial Day to Labor Day in the years 2002 and 2003. This is built as an expression of the intersection between 3 sets:

- Every other tuesday

© ISO 2007 – All rights reserved

- The years 2002 and 2003
- The season between Memorial Day and Labor Day

Example 80.

```
<example xsi:type="QSI_TS">
  <!-- intersection, because it's a QSI -->

  <!-- every other Tuesday -->
  <term xsi:type='PIVL_TS' alignment='DW'>
    <phase lowClosed='true' highClosed='false'>
      <low value='20001202'/>
      <high value='20001203'/>
    </phase>
    <period value='2' unit='wk'/>
  </term>

  <!-- 2002 and 2003 -->
  <term xsi:type='IVL_TS' lowClosed='true' highClosed='false'>
    <low value='20020101'/>
    <high value='20040101'/>
  </term>

  <!-- season between Memorial Day and Labor Day -->
  <!-- periodic hull between Memorial day and Labor Day -->
  <term xsi:type='QSP_TS'>
    <first type="QSI_TS">
      <!-- memorial day: intersection of mondays and last week of May -->
      <term xsi:type='PIVL_TS'>
        <phase highClosed='false'>
          <low value='19870525'/>
          <high value='19870601'/>
        </phase>
        <period value='1' unit='a'/>
      </term>
      <term xsi:type='PIVL_TS'>
        <phase highClosed='false'>
          <low value='19870105'/>
          <high value='19870106'/>
        </phase>
        <period value='1' unit='wk'/>
      </term>
    </first>
    <second type="QSI_TS">
      <!-- labor day :  intersection of mondays and first week of Sept -->
      <term xsi:type='PIVL_TS'>
        <phase highClosed='false'>
          <low value='19870901'/>
          <high value='19870908'/>
        </phase>
        <period value='1' unit='a'/>
      </term>
      <term xsi:type='PIVL_TS'>
        <phase highClosed='false'>
          <low value='19870105'/>
          <high value='19870106'/>
        </phase>
        <period value='1' unit='wk'/>
      </term>
    </second>
  </term>
</example>
```


© ISO 2007 – All rights reserved

7.11.2 QSU (QSET Union)

7.11.2.1 Description

Specializes QSET

Specifies a QSET as a union of other sets.

7.11.2.2 ISO 11404 Syntax:

```
type QSU (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  terms : Set(QSET(T))
)
```

7.11.2.3 Attributes

terms : Set(QSET(T)): a list of other QSETs that are involved in the union.

7.11.2.4 Invariants

- A nonNull QSU may only contain nonNull QSETs
- At least 2 sets must be specified.

OCL for invariants:

```
inv "not null": isNotNull implies terms->forAll(t |
  t.isNotNull)
inv "size": isNotNull implies terms->size >= 2
```

7.11.3 QSI (QSET Intersection)

7.11.3.1 Description

Specializes QSET

Specifies a QSET as an intersection of other sets.

7.11.3.2 11404 Syntax:

```
type QSI (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
```

© ISO 2007 – All rights reserved

```
      controlActExtension : characterstring,
      nullFlavor : NullFlavor,
      updateMode : UpdateMode,
      flavorId : characterstring,
      originalText : ED.TEXT,
      terms : Set(QSET(T))
   )
```

7.11.3.3 Attributes

7.11.3.3.1 terms : Set(QSET(T)): A list of other QSETs that are involved in the intersection.

7.11.3.4 Invariants

- A nonNull QSI may only contain nonNull QSETs
- At least 2 sets must be specified.

OCL for invariants:

```
inv "not null": isNotNull implies terms->forAll(t |
  t.isNotNull)
inv "size": isNotNull implies terms->size >= 2
```

7.11.4 QSD (QSET Difference)

7.11.4.1 Description

Specializes QSET

Specifies a QSET as the difference between two sets.

7.11.4.2 11404 Syntax:

```
    type QSU (T : ANY) = class (
      validTimeLow : characterstring,
      validTimeHigh : characterstring,
      controlActRoot : characterstring,
      controlActExtension : characterstring,
      nullFlavor : NullFlavor,
      updateMode : UpdateMode,
      flavorId : characterstring,
      originalText : ED.TEXT,
      first : QSET(T)
      second : QSET(T)
   )
```

The difference is the second set subtracted from the first.

© ISO 2007 – All rights reserved

7.11.4.3 Attributes

7.11.4.3.1 first : QSET(T): The set from which the second set is subtracted.

7.11.4.3.2 second: QSET(T): The set that is subtracted from the first set.

7.11.4.4 Invariants

- A nonNull QSD may only contain nonNull QSETs

OCL for invariants:

```
inv inv "not null": isNotNull implies (first.isNotNull and
  second.isNotNull)
```

7.11.5 QSP (QSET Periodic Hull)

7.11.5.1 Description

Specializes QSET

Specifies a QSET as the periodic hull between two sets.

7.11.5.2 ISO 11404 Syntax:

```
type QSP (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  first : QSET(T)
  second : QSET(T)
)
```

7.11.5.3 Attributes

7.11.5.3.1 first : QSET(T): The set used as the basis for the periodic hull operation.

7.11.5.3.2 second: QSET(T): The set that is used as the parameter for the periodic hull operation.

7.11.5.4 Invariants

- A nonNull QSP may only contain nonNull QSETs

OCL for invariants:

© ISO 2007 – All rights reserved

```
inv inv "not null": isNotNull implies (first.isNotNull and
  second.isNotNull)
```

7.11.6 QSS (QSET Set)

7.11.6.1 Description

Specializes QSET

Specifies a QSET as an enumeration of simple values. This is a shortcut form for specifying the same values as precise ranges.

7.11.6.2 ISO 11404 Syntax:

```
type QSS (T : QTY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  terms : Set(T)
)
```

7.11.6.3 Attributes

terms : Set(T): a list of values that are in the set. The set is actually constructed as the union of the intervals implied by the precision implicit in the definition of T. For some types of QTY, this is either trivial (INT) or ambiguous (RTO) and QSS doesn't really make sense for these type. QSS is a useful type for TS in particular.

7.11.6.4 Invariants

- A nonNull QSS may only contain nonNull values
- At least 1 value must be specified.

OCL for invariants:

```
inv "not null": isNotNull implies terms->forAll(t |
   t.isNotNull)
inv "size": isNotNull implies terms->size > 1
```

7.11.6.5 Examples

```
<example xsi:type='QSS_TS'>
   <term value='20071101'/>
   <term value='20071106'/>
</example>
```


© ISO 2007 – All rights reserved

The union of the intervals that cover the 1st November 2007 and the 6th November 2007.

7.11.7 QSC (Coded QSET)

7.11.7.1 Description

Specializes QSET

Specifies a QSET as an coded value that describes a predefined QSET(TS).

7.11.7.2 ISO 11404 Syntax:

```
type QSC (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  code : GTSAbbreviation
)
```

7.11.7.3 Attributes

code : GTSAbbreviation: a predefined code that describes the set.

If populated, the value of this attribute SHALL be taken from the HL7 GTSAbbreviation code system. The current values are:

Enumeration GTSAbbreviation. OID: 2.16.840.1.113883.5.1022		
1	AM	Every morning at institution specified times.
1	PM	Every afternoon at institution specified times.
1	BID	Two times a day at institution specified time
1	TID	Three times a day at institution specified time
1	QID	Four times a day at institution specified time
1	JB	Regular business days (Monday to Friday excluding holidays)
1	JE	Regular weekends (Saturday and Sunday excluding holidays)
1	JH	Holidays
2	JHCHR	Christian Holidays (Roman/Gregorian "Western" Tradition.)
3	JHCHRXME	Christmas Eve (December 24)
3	JHCHRXMS	Christmas Day (December 25)
3	JHCHRNEW	New Year's Day (January 1)
3	JHCHREAS	Easter Sunday. The Easter date is a rather complex calculation

© ISO 2007 – All rights reserved

		based on Astronomical tables describing full moon dates. Details can be found at [http://www.assa.org.au/edm.html, and http://aa.usno.navy.mil/AA/faq/docs/easter.html]. Note that the Christian Orthodox Holidays are based on the Julian calendar.
3	JHCHRGFR	Good Friday, is the Friday right before Easter Sunday
3	JHCHRPEN	Pentecost Sunday, is seven weeks after Easter (the 50th day of Easter).
2	JHNUS	United States National Holidays (public holidays for federal employees established by U.S. Federal law 5 U.S.C. 6103.)
3	JHNUSMLK	Dr. Martin Luther King, Jr. Day, the third Monday in January.
3	JHNUSPRE	Washington's Birthday (Presidential Day) the third Monday in February
3	JHNUSMEM	Memorial Day, the last Monday in May
3	JHNUSMEM5	Friday before Memorial Day Weekend
3	JHNUSMEM6	Saturday of Memorial Day Weekend
3	JHNUSIND	Independence Day (4th of July)
3	JHNUSIND5	Alternative Friday before 4th of July Weekend [5 U.S.C. 6103(b)].
3	JHNUSIND1	Alternative Monday after 4th of July Weekend [5 U.S.C. 6103(b)].
3	JHNUSLBR	Labor Day, the first Monday in September.
3	JHNUSCLM	Columbus Day, the second Monday in October.
3	JHNUSVET	Veteran's Day, November 11
3	JHNUSTKS	Thanksgiving Day, the fourth Thursday in November
3	JHNUSTKS5	Friday after Thanksgiving

Notes:

- This table is not complete, nor does it include religious holidays other than Christian (of the Gregorian [Western] tradition) or national holidays countries other than those of the US. This is a limitation to be remedied by subsequent additions. *Submissions are solicited as part of the balloting process.*

- Holidays are locale-specific. Exactly which religious holidays are subsumed under JH depends on the locale and other tradition. For global interoperability, using constructed QSET expressions is safer than named holidays. However, some holidays that depend on moon phases (e.g., Easter) or ad-hoc decree cannot be easily expressed in a QSET other than by use *QSC*.

- Applications may only need to implement a subset of these codes, often based on the locality of implementation.

7.11.7.4 Invariants

- A code must be provided.

OCL for invariants:

© ISO 2007 – All rights reserved

```
inv "not null": isNotNull implies code.isNotNull
```

7.11.7.5 Examples

```
<example xsi:type='QSC_TS' code=" JHCHRXMS"/>
```

All Christmas days.

7.11.8 IVL (Interval)

7.11.8.1 Description

Specializes QSET

Parameter: T : QTY

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While *IVL*s between 2 and 4 meter exists, there is no *IVL* between 2 meters and 4 seconds.

7.11.8.2 11404 Syntax:

```
type IVL (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  low : T,
  lowClosed : boolean,
  high : T,
  highClosed : boolean,
  width : QTY,
  any : T
)
```

7.11.8.3 Attributes

7.11.8.3.1 low : T: This is the low limit. If the low limit is not known, a nullFlavor may be specified.

The low limit SHALL NOT be positive infinity.

© ISO 2007 – All rights reserved

7.11.8.3.2 lowClosed : Boolean: Specifies whether low is included in the IVL (is closed) or excluded from the IVL (is open).

7.11.8.3.3 high : T: This is the high limit. If the high limit is not known, a nullFlavor may be specified.

The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

7.11.8.3.4 highClosed : Boolean: Specifies whether high is included in the IVL (is closed) or excluded from the IVL (is open).

7.11.8.3.5 width : QTY: The difference between high and low boundary. Width is used when the size of the Interval is known, but the actual start and end points are not known. The actual type QTY will be dictated by the type of T.

7.11.8.3.6 any : T: Specifies that nothing is known about the interval except that some particular value lies within the interval.

This should be used when it is not known when something started, or will end, but it is known that it was happening at a given time. This is relatively common for observations (i.e. of disease processes), procedure, and scheduling. If *any* has a value, then nothing else is known about the interval, except that the interval contains any.

7.11.8.4 Equality

Unlike other QSET specializations, IVL equality is determined based on set membership. Two IVL values are equal if they contain the same members.

Note 1: For IVLs, there are two special cases. If the upper limit of both IVLs is known to be positive infinity, or the lower limit is known to be negative infinity, then the SETs have the same elements.

Note 2: if two intervals have the same width and the bounds are not known, they are not considered equal.

Note 3: The same applies where the interval is known by a contained value (any): such intervals are never considered equal.

Note 4: Because equality is determined by set membership, it is possible for DSET(INT) and IVL(INT) to be equal. For example, the SET(INT) 2,3,4 is equal to the IVL(INT) 2..4.

7.11.8.5 Invariants

- Either the IVL is null, has a width, any, or has (low and/or high). Any, width and (low and/or high) cannot be mixed
- lowClosed and highClosed can only be used if low or high are used

OCL for Invariants:

```
def: let hasBounds : Boolean = low.isNotNull or
   high.isNotNull
inv "null rules": isNull xor (hasBounds or
   any.isNotNull or width.isNotNull)
```


© ISO 2007 – All rights reserved

```
inv "co-occurence rules": isNotNull implies (
   (any.isNotNull implies not (hasBounds or width.isNotNull))
   and (any.isNull implies (hasBounds xor width.isNotNull)))
inv "closed attributes only if limited":
     (not low.isNotNull implies lowClosed.oclIsUndefined) and
     (not high.isNotNull implies highClosed.oclIsUndefined)
inv "no updateMode or History on IVL attributes":
    noUpdateOrHistory(low) and noUpdateOrHistory(high) and
    noUpdateOrHistory(width)
```

7.11.8.6 Examples

7.11.8.6.1 Integer interval

```
<example xsi:type='IVL_INT'>
   <low value='2'/>
   <high value='4'/>
</example>
```

A simple interval of INT between 2 and 4. This is exactly the same set of values as specifying the DSET (2,3,4)

7.11.8.6.2 Physical Quantity Interval

```
<example xsi:type='IVL_PQ' lowClosed='true' highClosed='false'>
   <low value='2.8' unit='m'/>
   <high value='4.6' unit='m'/>
</example>
```

An interval of PQ between 2.8 meters, inclusive, and 4.6 meters, exclusive.

7.11.8.6.3 Timestamp Interval

```
<example xsi:type='IVL_TS'>
   <low value='200012041000'/>
   <high value='200012041030'/>
</example>
```

An interval of TS on December 4, 2000, between 10:00 am and 10:30 am.

7.11.9 IVL.LOW

7.11.9.1 Description

Constrains IVL

IVL.LOW constrains IVL so that low is provided and lowClosed is true. All other properties are prohibited.

7.11.9.2 Invariants

- low and lowClosed must be populated
- high and highClosed must be null

OCL for Invariants:

© ISO 2007 – All rights reserved

```
inv "low": isNotNull implies low.isNotNull and lowClosed
inv "high": high.oclIsUndefined and highClosed.oclIsUndefined
```

7.11.10 IVL.HIGH

7.11.10.1 Description

Constrains IVL

IVL.HIGH constrains IVL so that high is provided and highClosed is true. All other properties are prohibited.

7.11.10.2 Invariants

- low and lowClosed must be null
- high and highClosed must be populated

OCL for Invariants:

```
inv "high": isNotNull implies high.isNotNull and highClosed
inv "low": low.oclIsUndefined and lowClosed.oclIsUndefined
```

7.11.11 IVL.WIDTH

7.11.11.1 Description

Constrains IVL

IVL.WIDTH constrains IVL so that Width is mandatory and low, lowClosed, high and highClosed are prohibited.

7.11.11.2 Invariants

- Width must be populated
- low and lowClosed must be null
- high and highClosed must be null

OCL for Invariants:

```
inv "width": isNotNull implies width.isNotNull
inv "low": low.oclIsUndefined and lowClosed.oclIsUndefined
inv "high": high.oclIsUndefined and highClosed.oclIsUndefined
```

7.11.12 PIVL (PeriodicInterval)

7.11.12.1 Description

Specializes QSET

© ISO 2007 – All rights reserved

An interval of time that recurs periodically. PIVL has two properties, phase and period/frequency. phase specifies the "interval prototype" that is repeated on the period/frequency.

7.11.12.2 ISO 11404 Syntax:

```
type PIVL (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  phase : IVL(TS),
  period : PQ,
  frequency RTO(INT,PQ.TIME),
  alignment : CalendarCycle,
  isFlexible : boolean,
  count : INT.POS
)
```

7.11.12.3 Attributes

7.11.12.3.1 phase : IVL(TS): A prototype of the repeating interval, specifying the duration of each occurrence and anchors the PIVL sequence at a certain point in time. phase also marks the anchor point in time for the entire series of periodically recurring intervals. If count is null, the recurrence of a PIVL has no beginning or ending, but is infinite in both future and past.

The width of the phase SHALL be less than or equal to the period.

7.11.12.3.2 period : PQ: A time duration specified as a reciprocal measure of the frequency at which the PIVL repeats.

7.11.12.3.3 frequency : RTO: The number of times the *PIVL* repeats (numerator) within a specified time-period (denominator). The numerator is an integer, and the denominator is a PQ.TIME.

Note: Only one of period and frequency should be specified. The form chosen should be the form that most naturally conveys the idea to humans. i.e. Every 10 mins (period) or twice a day (frequency).

7.11.12.3.4 alignment : CalendarCycle: Specifies if and how the repetitions are aligned to the cycles of the underlying calendar (e.g., to distinguish every 30 days from "the 5th of every month".) A non-aligned PIVL recurs independently from the calendar. An aligned PIVL is synchronized with the calendar.

If populated, the value of this attribute SHALL be taken from the HL7 CalendarCycle code system. The current values are:

CalendarCycle Enumeration. OID: 2.16.840.1.113883.5.9			
1	CY	year	
1	MY	month of the year	

© ISO 2007 – All rights reserved

1	CM	month (continuous)	
1	CW	week (continuous)	
1	WY	week of the year	
1	DM	day of the month	
1	CD	day (continuous)	
1	DY	day of the year	
1	DW	day of the week (begins with monday)	
1	HD	hour of the day	
1	CH	hour (continuous)	
1	NH	minute of the hour	
1	CN	minute (continuous)	
1	SN	second of the minute	
1	CS	second (continuous)	

ISO 11404 Syntax:

```
 type CalendarCycle = enumeration (CY, MY, CM, CW, WY, DM,
CD, DY, DW, HD, CH, NH, CN, SN, CS)
```

7.11.12.3.5 isFlexible : Boolean: Indicates whether the exact timing is up to the party executing the schedule e.g., to distinguish "every 8 hours" from "3 times a day".

7.11.12.3.6 count : INT.POS: The number of times the period repeats in total. If count is null, then the period repeats indefinitely both before and after the anchor implicit in the phase.

7.11.12.4 Invariants

- If PIVL is not null, phase and (period or frequency) must be specified
- If PIVL is not null, only one of period and frequency may be specified

OCL for invariants:

```
inv "period or frequency": isNotNull implies
inv(period.isNotNull xor frequency.isNotNull)
inv "no updates on attributes":
     noUpdateOrHistory(phase)and noUpdateOrHistory(period)
```

7.11.12.5 Examples

7.11.12.5.1 Twice a day

```
<example xsi:type='PIVL_TS' isFlexible='true'>
   <period value='12' unit='h'/>
</example>
```

Twice a day (BID). The actual time is at the discretion of the institution.

© ISO 2007 – All rights reserved

7.11.12.5.2 Twice a day for ten minutes

```
<example xsi:type='PIVL_TS'>
   <phase>
      <width xsi:type="PQ" value='10' unit='min'/>
   </phase>
   <period value='12' unit='h'/>
</example>
```

Twice a day (every twelve hours) for ten minutes.

7.11.12.5.3 Every September

```
<example xsi:type='PIVL_TS' alignment='MY'>
   <phase highClosed='true' lowClosed='false'>
      <low value='198709'/>
      <high value='198710'/>
   </phase>
   <period value='1' unit='a'/>
</example>
```

This example is slightly more complex and shows the month of September that recurs every year (note that in the year 1987 in this form is irrelevant since the periodic interval recurs every year past and future.)

7.11.12.5.4 Every other Saturday.

```
<example xsi:type='PIVL_TS' alignment='DW'>
   <phase highClosed='true' lowClosed='false'>
      <low value='20001202' />
      <high value='20001203' />
   </phase>
   <period value='2' unit='wk'/>
</example>
```

7.11.12.5.5 Every four to six hours.

```
<example xsi:type='PIVL_TS'>
   <period value='5' unit='h' uncertaintyType='U'>
      <uncertainty value='0.57735' unit='h'/>
   </period>
</example>
```

7.11.13 EIVL (Event-Related Periodic Interval of Time)

7.11.13.1 Description

Specializes QSET

Specifies a periodic interval of time where the recurrence is based on activities of daily living or other important events that are time-related but not fully determined by time.

Example: "one hour after breakfast" specifies the beginning of the interval at one hour after breakfast is finished. Breakfast is assumed to occur before lunch but is not determined to occur at any specific time.

© ISO 2007 – All rights reserved

7.11.13.2 ISO 11404 Syntax:

```
type EIVL = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  originalText : ED.TEXT,
  event : TimingEvent,
  offset : IVL(PQ)
)
```

7.11.13.3 Attributes

7.11.13.3.1 event : TimingEvent: A code for a common (periodic) activity of daily living based on which the event related periodic interval is specified. Events that qualify for being adopted in the domain of this attribute must satisfy the following requirements:

- the event commonly occurs on a regular basis
- the event is being used for timing activities, and
- the event is not entirely determined by time

If populated, the value of this attribute SHALL be taken from the HL7 TimingEvent code system. The current values are:

TimingEvent Enumeration. OID: 2.16.840.1.113883.5.139			
1	HS	HS	the hour of sleep
1	AC	AC	before meal (from lat. ante cibus)
2	ACM	ACM	before breakfast (from lat. ante cibus matutinus)
2	ACD	ACD	before lunch (from lat. ante cibus diurnus)
2	ACV	ACV	before dinner (from lat. ante cibus vespertinus)
1	IC	IC	between meals (from lat. inter cibus)
2	ICM	ICM	between breakfast and lunch
2	ICD	ICD	between lunch and dinner
2	ICV	ICV	between dinner and the hour of sleep
1	PC	PC	after meal (from lat. post cibus)
2	PCM	PCM	after breakfast (from lat. post cibus matutinus)
2	PCD	PCD	after lunch (from lat. post cibus diurnus)
2	PCV	PCV	after dinner (from lat. post cibus vespertinus)

ISO 11404 Syntax:

© ISO 2007 – All rights reserved

```
type TimingEvent = enumeration (C, CD, CM, CV, HS,
COIT, DIAP, DEFEC)
```

7.11.13.3.2 offset : IVL(PQ): An interval of elapsed time (duration, not absolute point in time) that marks the offsets for the beginning, width and end of the EIVL measured from the time each such event actually occurred.

Example: if the specification is "one hour before breakfast for 10 minutes" IVL.low of offset is -1 h and the IVL.high of offset is -50 min.

The interval SHALL be consistent with the event code. If the event indicates that the event related interval occurs before the event, the interval low and high bounds SHALL be both negative or null. If the event indicates the event related interval occurs after the event, then the interval low and high bounds SHALL both be positive or null. If the event indicates that the event related interval occurs between two events, then the interval bounds SHALL be null; i.e. the interval must have a width only.

7.11.13.4 Invariants

- If EIVL is not null, event must be specified

OCL for invariants:

```
inv "required attributes": isNotNull implies
   (event.oclIsDefined)
inv "no updateMode or History on EIVL attributes":
    noUpdateOrHistory(offset)
```

7.11.13.5 Examples

```
<example xsi:type='EIVL_TS' event='CM'/>
   <offset>
      <low value='-1' unit='h'/>
      <high value='-50' unit='min'/>
   </offset>
</example>
```

One hour before breakfast for 10 minutes.

Example 75.

```
<example xsi:type='EIVL_TS' event='CV'/>
   <offset>
      <low value='30' unit='min'/>
      <high value='30' unit='min'/>
   </offset>
</example>
```

30 minutes after dinner.

7.11.14 GTS.BOUNDEDPIVL

7.11.14.1 Description

Constrains QSI

© ISO 2007 – All rights reserved

GTS.BOUNDEDPIVL constrains QSI(TS) so that it only allows an intersection of IVL(TS) and PIVL(TS).

7.11.14.2 Invariants

- low and lowClosed must be null
- high and highClosed must be populated

OCL for Invariants:

```
inv "QSET.BOUNDEDPIVL 2": terms->size = 2
inv "GTS.BoundedPIVL 3": terms.item->exists(t |
    t.oclIsKindOf(IVL(TS)))
inv "GTS.BoundedPIVL 3": terms.item->exists(t |
    t.oclIsKindOf(PIVL(TS)))
```


© ISO 2007 – All rights reserved

7.12 Uncertainty Datatypes

These data types provide support for uncertain values. Note the support provided here, along with the support provided for uncertainty on QTY, provides support for quantitative uncertainty, not with the medical kinds of uncertainty encountered in clinical practice such as “likely to be x”, or a differential diagnoses.

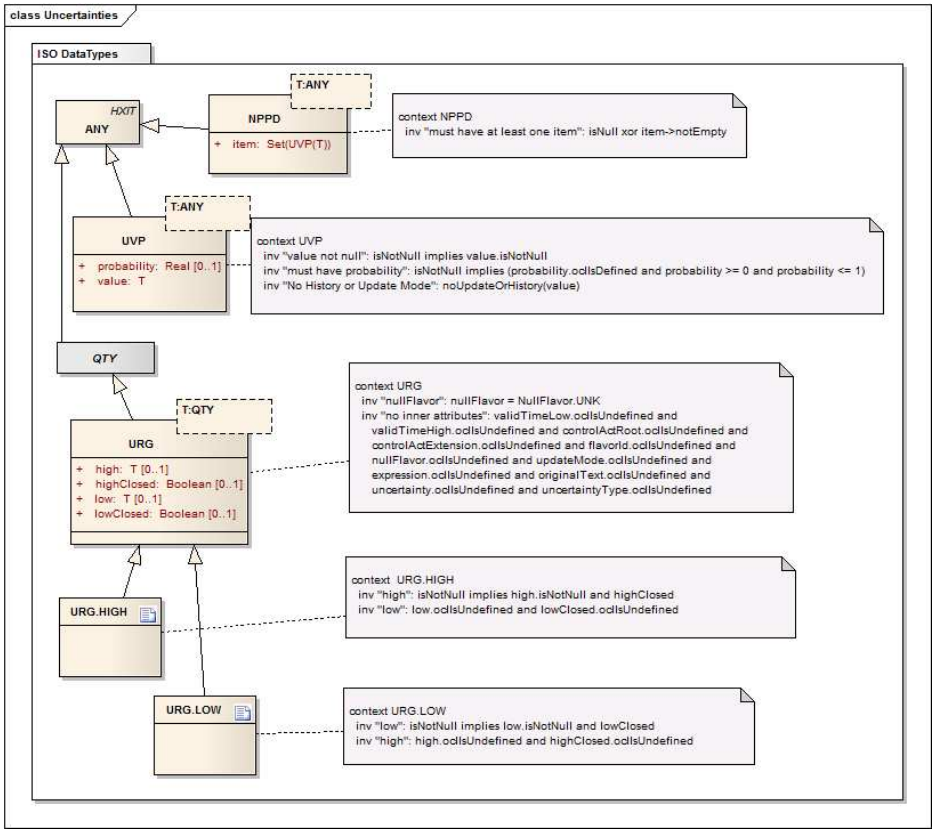


Figure 12. Uncertainty data types

7.12.1 UVP (Uncertain Value - Probabilistic)

7.12.1.1 Description

Specializes ANY

Parameter: T : ANY

A generic datatype extension used to specify a probability expressing the information producer's belief that the given value holds.

7.12.1.2 11404 Syntax:

```
type UVP (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
```

© ISO 2007 – All rights reserved

```
      updateMode : UpdateMode,
      flavorId : characterstring,
      probability : real,
      value : T
    )
```

7.12.1.3 Attributes

7.12.1.3.1 probability : real: The probability assigned to the value, a decimal number between 0 (very uncertain) and 1 (certain), inclusive.

There is no "default probability" that one can assume when the probability is unstated. Therefore, it is impossible to make any semantic difference between a UVP without probability and a simple T. UVP does not mean "uncertain", and a simple T does not mean "certain". In fact, the probability of the UVP could be 0.999 or 1, which is quite certain, where a simple T value could be a very vague guess.

7.12.1.3.2 value : T: The value of T which the probability refers to.

7.12.1.4 Equality

Two nonNull UVP values are equal if the probability and value are equal.

7.12.1.5 Invariants

- a value must be provided
- a probability must be provided
- probability must be between 0 and 1

OCL for Invariants:

```
  inv"value not null": isNotNull implies value.isNotNull

  inv "must have probability": isNotNull implies
       (probability.oclIsDefined and probability >= 0
       and probability <= 1)
  inv "No History or Update Mode": noUpdateOrHistory(probability)
       and noUpdateOrHistory(value)
```

7.12.2 NPPD (Non-Parametric Probability Distribution)

7.12.2.1 Description

Specializes ANY

Parameter: T : ANY

A set of UVP with probabilities (also known as a histogram.) All the elements in the set are considered alternatives and are rated each with its probability expressing the belief (or frequency) that each given value holds.

© ISO 2007 – All rights reserved

7.12.2.2 11404 Syntax:

```
type NPPD (T : ANY) = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  item : Set(UVP(T))
)
```

7.12.2.3 Attributes

7.12.2.3.1 item : Set(UVP(T)): The list of values with probabilities for the histogram.

7.12.2.4 Equality

Two nonNull NPPDs are equal if they contain the same elements.

Note: the determination of element content is based on the same semantic equals as defined in this specification, so it is possible that a NPPD(CD) can be equal to NPPD(CS), for instance.

7.12.2.5 Invariants

- at least one value must be provided

OCL for Invariants:

```
inv "must have at least one item": isNull xor item->notEmpty
```

7.12.2.6 Examples

```
<example xsi:type='NPPD_ST'>
  <item probability="0.1">
        <value value="Yankees"/>
  </item>
  <item probability="0.04">
        <value value="Red Sox"/>
  </item>
  <item probability="0.05">
        <value value="White Sox"/>
  </item>
  <item probability="0.08">
        <value value="Indians"/>
  </item>
  <item probability="0.05">
        <value value="Tigers"/>
  </item>
  <item probability="0.07">
        <value value="Mariners"/>
  </item>
  <item probability="0.02">
        <value value="Royals"/>
```

© ISO 2007 – All rights reserved

```
  </item>
  <item probability="0.06">
        <value value="Orioles"/>
  </item>
</example>
```

7.12.3 URG (Uncertain Range)

7.12.3.1 Description

Specializes QTY

Parameter: T : QTY

Indicates that the value comes from a range of possible values of the type T

Note 1: URG is used where the actual value is unknown, but it is known that the value comes from a known range of possible values. URG differs from QTY.uncertainty in that QTY.uncertainty is used to report a particular value along with an associated distribution of uncertainty for the value, or to report the summary distribution of a set of data, whereas URG indicates that there is a single value that, although unknown, comes from a particular range of values. No inference regarding distribution of values can be taken. URG is often associated with an instruction to perform a particular operation at some point within a given time interval.

Note 2: Any ordered type can be the basis of an URG; it does not matter whether the base type is discrete or continuous. If the base data type is only partially ordered, all possible values for the URG are elements of a totally ordered subset of the partially ordered data type. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While URGs between 2 and 4 meter exists, there is no URG between 2 meters and 4 seconds.

Note 3: Implicit in the use of URG is a statement that the actual value is unknown.

7.12.3.2 ISO 11404 Syntax:

```
type URG = class (
  validTimeLow : characterstring,
  validTimeHigh : characterstring,
  controlActRoot : characterstring,
  controlActExtension : characterstring,
  nullFlavor : NullFlavor,
  updateMode : UpdateMode,
  flavorId : characterstring,
  expression : ED,
  originalText : ED.TEXT,
  uncertainty : QTY,
  uncertaintyType : UncertaintyType,
  low : T,
  lowClosed : boolean,
  high : T,
  highClosed : boolean,
)
```


© ISO 2007 – All rights reserved

7.12.3.3 Attributes

7.12.3.3.1 low : T: This is the low limit. If the low limit is not known, a nullFlavor may be specified.

The low limit SHALL NOT be positive infinity.

7.12.3.3.2 lowClosed : Boolean: Specifies whether low is included in the URG (is closed) or excluded from the URG (is open).

7.12.3.3.3 high : T: This is the high limit. If the high limit is not known, a nullFlavor may be specified.

The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

7.12.3.3.4 highClosed : Boolean: Specifies whether high is included in the URG (is closed) or excluded from the URG (is open).

7.12.3.4 Equality

It is never known whether values or type URG(T) are equal with each other, since the actual value is unknown.

7.12.3.5 Invariants

- NullFlavor is fixed to UNK
- No inherited semantics on the contained type (all must be on the URG).

OCL for Invariants:

```
inv "nullFlavor": nullFlavor = UNK
inv "no inner attributes": validTimeLow.oclIsUndefined and
 validTimeHigh.oclIsUndefined and controlActRoot.oclIsUndefined and
 controlActExtension.oclIsUndefined and flavorId.oclIsUndefined and
 nullFlavor.oclIsUndefined and updateMode.oclIsUndefined and
 expression.oclIsUndefined and originalText.oclIsUndefined and
 uncertainty.oclIsUndefined and uncertaintyType.oclIsUndefined
```

7.12.3.5.1 Operations

7.12.3.5.2 couldBe (other : T) : BL: Although the actual value is unknown, it is possible to determine whether a URG could be a particular value. A particular value is possible if it is contained within the specified range..

7.12.3.6 Example

```
<doseQuantity xsi:type="URG_PQ">
  <low value="1"/>
  <high value="2"/>
</doseQuantity>
```

© ISO 2007 – All rights reserved

This URG(PQ) specifies that patient should take 1-2 tablets. This might be part of a prescription order such as "By mouth, take 1-2 tablets every 4-6 hours when needed for severe pain to a maximum of 8 per day".

7.12.4 URG.LOW

7.12.4.1 Description

Constrains URG

URG.LOW constrains URG so that low is required and lowClosed is true. All other properties are prohibited.

7.12.4.2 Invariants

- low and lowClosed must be populated
- high and highClosed must be null

OCL for Invariants:

```
inv "low": low.isNotNull and lowClosed
inv "high": high.oclIsUndefined and highClosed.oclIsUndefined
```

7.12.5 URG.HIGH

7.12.5.1 Description

Constrains URG

URG.HIGH constrains IVL so that high is provided and highClosed is true. All other properties are prohibited.

7.12.5.2 Invariants

- low and lowClosed must be null
- high and highClosed must be populated

OCL for Invariants:

```
inv "high": high.isNotNull and highClosed
inv "low": low.oclIsUndefined and lowClosed.oclIsUndefined
```


© ISO 2007 – All rights reserved

7.13 Structured Text

This section documents the inner structure of the ED.STRUCTUREDTEXT and ED.STRUCTUREDTITLE data type flavors.

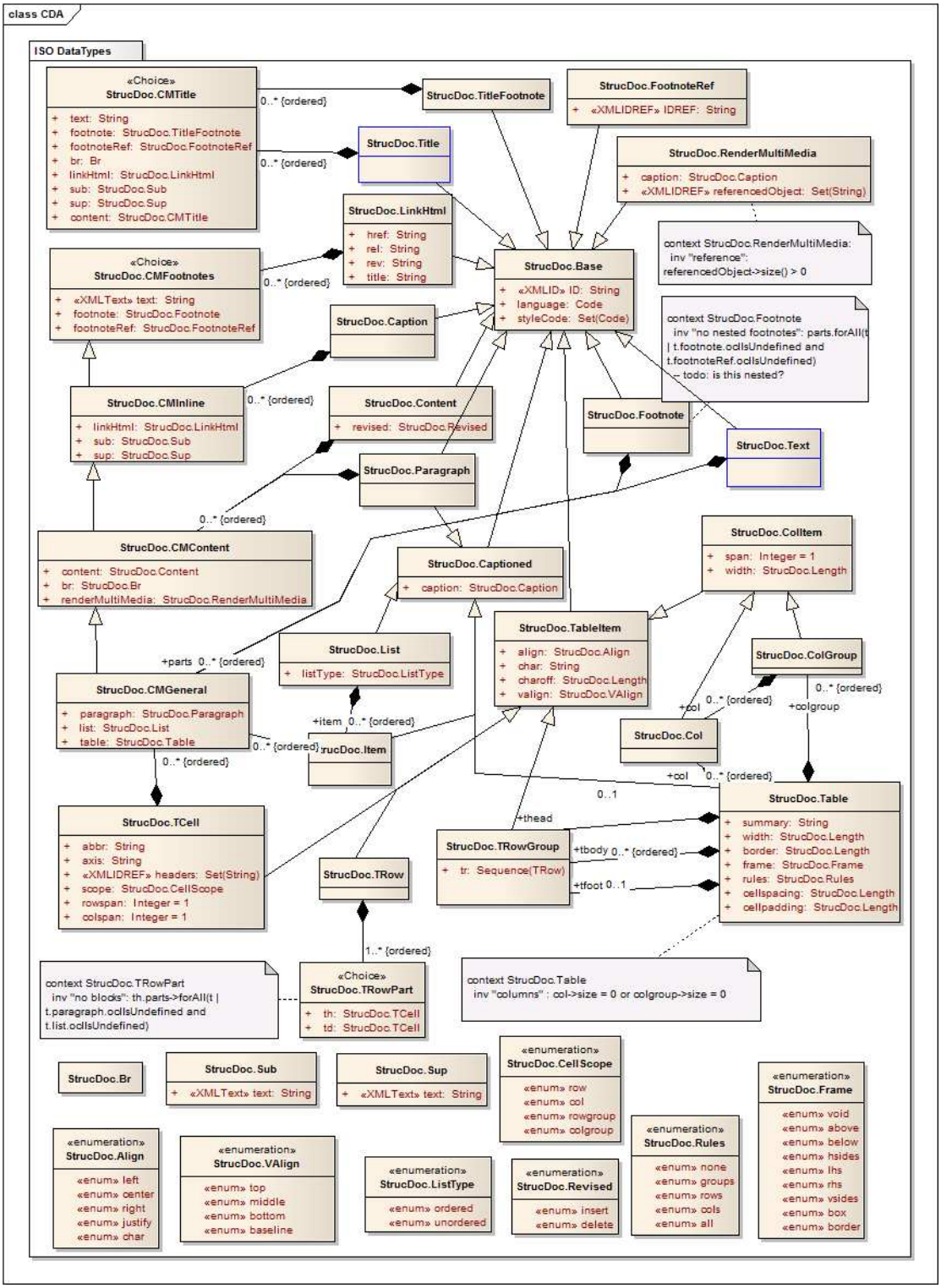


Figure 13. Structured Text Model

© ISO 2007 – All rights reserved

Both ED.STRUCTUREDTEXT and ED.STRUCTUREDTITLE contain documents. In addition to plain text, the documents may contain formatting, and referencing information. Structured text documents may contain multi-media content, lists and tables.

These document are not intended to be complete documents. They are building blocks that are intended to be integrated into a larger context with other classes that use these document formats with other datatypes to build a complete document that is useful for clinical or other healthcare use. However these document formats may also find a use inside other record and messages formats. Whether documents, records, or messages, these documents will be created within a single context. This document context may contain multiple documents of the type defined in this section, and the documents may refer to any content within this document context.

Any Information Processing Entity claiming direct or indirect conformance to this standard that supports the use of the document formats defined in this specification SHALL document the scope of the context and clearly define how references within the document context are resolved.

There is special support in this section for relating content changes to previous versions of the content. If it is supported, the conformance statement SHALL make it clear how the applicable version information is made available and integrated with the change control content,.

These documents are intended to be "rendered". A document is rendered when it is prepared for human consumption, either on a computer screen or a printed report or by some other method. However the document is rendered, the rules described in this section SHALL be followed.

7.13.1.1 Example

```
<example>
  <paragraph>
    <caption styleCode="Bold xHead1">Introduction</caption>
    Thank you for referring this patient for investigation
    into <content ID="c1">burnt ears</content>.
  </paragraph>
  <list>
    <caption styleCode="Bold xHead1">Initial Observations</caption>
    <item>The patient presented in a very confused state.</item>
    <item>
       There was extensive damage to the outer ears:
       <renderMultiMedia referencedObject="i1">
         <caption>Photo of left ear</caption>
       </renderMultiMedia>
    </item>
  </list>
  <table summary="Investigations performed"  border="all" rules="all">
      <caption styleCode="Bold xHead1">Investigations</caption>
    <thead>
      <tr>
        <th>Investigation</th><th>Finding</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <th><content ID="c2">Skin Condition</content></th>
        <th><content ID="c3">1<sup>st</sup> degree burns</content></th>
      </tr>
      <tr>
        <th><content ID="c4">Hearing Test</content></th>
```


© ISO 2007 – All rights reserved

```
        <th><content ID="c5">The patients hearing is okay</content></th>
      </tr>
    </tbody>
  </table>
  <paragraph>
    <caption styleCode="Bold xHead1">Recommendations</caption>
      The patient should apply a cream to the outer ears until
      they are healed. <content revised="insert">The patient
      should wear a woollen balaclava in the future while ironing
      his shirts to prevent are-occurence of the accident<footnote>This
      has been proven to offer the best protection against a repeat
      injury. See <linkHtml href="http://www.wikipedia.org/wiki/burnt_ears">
      the wikipedia article about burnt ears</linkHtml> for further
      information.</footnote>.</content>
  </paragraph>
</example>
```

Notes:

- This narrative has 4 sections: an introduction, initial observations, a summary of investigations, and recommendations. Besides not being a serious example, the context of use of a structured text like this may make specific rules about the semantic scope of a particular piece of structured text, thereby narrowing the scope of a particular structured text
- The styleCode xHead1 is an example of a valid local extension to the style codes
- The content element is used to introduce an ID attribute. This is suitable for use in an originalText.reference element on a CD or PQ type in other content that refers into the structured text
- The final use of the content element shows a revision to the document.
- There must be some other item with the ID "i1" in the scope of the document context. This should some kind of class that unambiguously provides some multiemedia content
- While the actual presentation process is outside of this specification, one possible presentation of the document would take this form:

Introduction

Thank you for referring this patient for investigation into burnt ears

Initial Observations

- The patient presented in a very confused state
- There was extensive damage to the outer ears:

Photo of left ear:

(Picture of ear)

Investigations

Investigation	Finding
Skin Condition	1st degree burns
Hearing Test	The patients hearing is okay

Recommendations

The patient should apply a cream to the outer ears until they are healed. The patient should wear a woollen balaclava in the future while ironing his shirts to prevent a re-occurence of the accident[1].

[1] This has been proven to offer the best protection against a repeat injury. See the wikipedia article about burnt ears for further information.

© ISO 2007 – All rights reserved

7.13.2 StrucDoc.Base

7.13.2.1 Description

Abstract.

Defines basic identificaition and styling attributes shared by many structured document elements.

7.13.2.2 Attributes

7.13.2.2.1 ID : String: Unique Identity of this element within a document.

7.13.2.2.2 language : Code: Language of the element. See Section 7.4.2.3.7 for more information. Within a document, the language applies to all the contained elements unless some other language is specifically identified.

7.13.2.2.3 styleCode : Set(Code): Styles that apply to this document.

If populated, the value of this attribute SHALL be taken from one of these values listed below or a valid local extension:

StyleCode Enumeration		
1		*Font style (Defines font rendering characteristics.)*
2	Bold	Render with a bold font.
2	Underline	Render with an underlines font.
2	Italics	Render italicized.
2	Emphasis	Render with some type of emphasis.
1		*Table rule style (Defines table cell rendering characteristics.)*
2	Lrule	Render cell with left-sided rule.
2	Rrule	Render cell with right-sided rule.
2	Toprule	Render cell with rule on top.
2	Botrule	Render cell with rule on bottom.
1		*Ordered list style (Defines rendering characteristics for ordered lists.)*
2	Arabic	List is ordered using Arabic numerals: 1, 2, 3.
2	LittleRoman	List is ordered using little Roman numerals: i, ii, iii.
2	BigRoman	List is ordered using big Roman numerals: I, II, III.
2	LittleAlpha	List is ordered using little alpha characters: a, b, c.
2	BigAlpha	List is ordered using big alpha characters: A, B, C.
1		*Unordered list style (Defines rendering characteristics for unordered lists.)*
2	Disc	List bullets are simple solid discs.

© ISO 2007 – All rights reserved

2	Circle	List bullets are hollow discs.
2	Square	List bullets are solid squares.

Local extensions to the styleType enumeration must follow the following convention: [x][A-Za-z][A-Za-z0-9]* (first character is "x", second character is an upper or lower case A-Z, remaining characters are any combination of upper and lower case letters or numbers).

7.13.3 StrucDoc.Br

7.13.3.1 Description

Definition

A hard line break, like in XHTML.

7.13.4 StrucDoc.Sup

7.13.4.1 Description

Indicates that the value text should be rendered as superscript when presented. i.e. x^2.

7.13.4.2 Attributes

7.13.4.2.1 text : String «XMLText»: The text that is to be represented as superscript.

Refer to section A.2 for the meaning of the XMLText stereotype.

7.13.4.3 Invariants

- Text must not be empty

OCL for Invariants:

```
inv "text must not be empty": text.length > 0
```

7.13.5 StrucDoc.Sub

7.13.5.1 Description

Indicates that the value text should be rendered as subscript when presented. i.e. H_2O

7.13.5.2 Attributes

7.13.5.2.1 text : String «XMLText»: The text that is to be represented as subscript.

Refer to section A.2 for the meaning of the XMLText stereotype.

© ISO 2007 – All rights reserved

7.13.5.3 Invariants

- Text must not be empty

OCL for Invariants:

```
inv "text must not be empty": text.length > 0
```

7.13.6 StrucDoc.LinkHtml

7.13.6.1 Description

Specializes StrucDoc.Base

A hypertext reference to another document. These links are generally shown as hyperlinks that a user may activate when viewing the document.

The link functionality provides a generic referencing mechanism, similar, but not identical, to the HTML anchor tag. It can be used to reference identifiers that are either internal or external to the document or the document context.

Note: Multimedia that is integral to a document must be referenced by the renderMultiMedia element (see below). Multimedia that is simply referenced by the document and not an integral part of the document can by provided by a link. There is no requirement that a receiver render an internal or external link, or the target of an external link.

7.13.6.2 Attributes

7.13.6.2.1 href : String: The URL that identifiers the target documen/object of the link. The target is an XML identifier either internal or external to the document.The context of use must clearly define the scope of resolution of the link. Following the conventions of HTML, an internal link – usually a link within the scope of the context of the document - is prefaced with the pound sign.

7.13.6.2.2 rel : Set(StrucDoc.LinkType): This attribute describes the relationship from the current document to the anchor specified by the href attribute. The value of this attribute is a space-separated list of link types.

If populated, the value of this attribute SHALL be taken from one of these values listed below:

LinkType Enumeration		
1	Alternate	Designates substitute versions for the document in which the link occurs.
1	Stylesheet	Refers to an external style sheet. This is used together with the link type "Alternate" for user-selectable alternate style sheets.
1	Start	Refers to the first document in a collection of documents.
1	Next	Refers to the next document in a linear sequence of documents. User agents may choose to preload the "next" document, to reduce the perceived load time.
1	Prev	Refers to the previous document in an ordered series of documents. Some user agents also support the synonym

© ISO 2007 – All rights reserved

		"Previous".
1	Contents	Refers to a document serving as a table of contents. Some user agents also support the synonym *ToC* (from "Table of Contents").
1	Index	Refers to a document providing an index for the current document.
1	Glossary	Refers to a document providing a glossary of terms that pertain to the current document.
1	Copyright	Refers to a copyright statement for the current document.
1	Chapter	Refers to a document serving as a chapter in a collection of documents.
1	Section	Refers to a document serving as a section in a collection of documents.
1	Subsection	Refers to a document serving as a subsection in a collection of documents.
1	Appendix	Refers to a document serving as an appendix in a collection of documents.
1	Help	Refers to a document offering help (more information, links to other sources information, etc.)
1	Bookmark	Refers to a bookmark. A bookmark is a link to a key entry point within an extended document. The `title` attribute may be used, for example, to label the bookmark. Note that several bookmarks may be defined in each document

This list is taken from the HTML specification.

7.13.6.2.3 rev : Set(StrucDoc.LinkType): This attribute is used to describe a reverse link [p.155] from the anchor specified by the href attribute to the current document. The value of this attribute is a space-separated list of link types.See above for valid values.

7.13.6.2.4 title : String: This attribute offers advisory information about the element for which it is set. The title attribute has an additional role when used with a LINK element that designates an external style sheet. Please consult the HTML standard for additional information.

Note: Values of the title attribute may be rendered by user agents in a variety of ways. For instance, visual browsers frequently display the title as a "tool tip" (a short message that appears when the pointing device pauses over an object). Audio user agents may speak the title information in a similar context. For example, setting the attribute on a link allows user agents (visual and non-visual) to tell users about the nature of the linked resource:

7.13.6.3 Associations

7.13.6.3.1 parts: CMFootnotes [0..* ordered] «Anonymous»: The content (text and footnotes) that represent the text with which the activatable link is associated.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

© ISO 2007 – All rights reserved

7.13.6.4 Invariants

- Some text must be associated with the link

OCL for Invariants:

```
inv "must have at least one item": parts->notEmpty
```

7.13.6.5 Example

```
   <text>History of coronary artery disease, as noted
      <linkHtml href="#SECT001">above</linkHtml>.
   </text>
```

An explicit reference. In this example, the reference is to a CDA section with the id "SECT001":

```
<section ID="SECT003">
   <code code="10153-2" codeSystem="2.16.840.1.113883.6.1"
      codeSystemName="LOINC"/>
   <title>Past Medical History</title>
</section>
```

7.13.7 StrucDoc.RenderMultiMedia

7.13.7.1 Description

Specializes StrucDoc.Base

References multimedia content that is integral to the document, and serves to show where the referenced multimedia is to be rendered. The multi media content must be contained within the context of the document.

There is an optional caption, and contains a required referencedObject attribute (of type XML IDREFS), the values of which must equal the XML ID value(s) of ObservationMedia or RegionOfInterest CDA entries within the document context.

7.13.7.2 Attributes

7.13.7.2.1 caption : StrucDoc.Caption: An optional caption for the multimedia content.

7.13.7.2.2 referencedObject : Set(String) «XMLIDREF» : The references are to other identified objects within the document context.

Refer to section A.2 for the meaning of the XMLIDREF stereotype.

7.13.7.3 Invariants

- at least one reference must be provided

OCL for Invariants:

© ISO 2007 – All rights reserved

```
inv "must have at least one ref": referencedObject->size() > 0
```

7.13.8 StrucDoc.FootnoteRef

7.13.8.1 Description

Specializes StrucDoc.Base

A reference to an existing footnote within the document context. This may be used when the same footnote is being used multiple times. The value of the footnoteRef.IDREF must be an footnote.ID value in the same document.

7.13.8.2 Attributes

7.13.8.2.1 IDREF: String «XMLIDREF»: The identity of the referenced footnote.

Refer to section A.2 for the meaning of the XMLIDREF stereotype.

7.13.8.3 Invariants

- at reference must be provided

OCL for Invariants:

```
inv "must have a reference": IDREF.oclIsDefined
```

7.13.9 StrucDoc.Footnote

7.13.9.1 Description

Specializes StrucDoc.Base

Indicates a footnote. The content contained within the Footnote is the content of the footnote. When the document is rendered, a link to the footnote is displayed inline with the flow of text adjacent to the footnote.

Note: Receivers are required to interpret these elements when rendering by visually distinguishing footnoted text. The exact rendition is at the discretion of the recipient, and might include a mark at the location of the footnote with a hyperlink to the footnoted text, a simple demarcation (such as "This is the text [this is the footnote] that is being footnoted"), etc.

7.13.9.2 Associations

7.13.9.2.1 parts : StrucDoc.CMGeneral[0..* ordered] «Anonymous»: The contents of the footnote.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

© ISO 2007 – All rights reserved

7.13.9.3 Invariants

- The footnote must contain some content
- Footnotes cannot contain nested footnotes

OCL for Invariants:

```
inv "Some content required": parts->notEmpty
inv "no nested footnotes": parts->forAll(t |
   t.footnote.oclIsUndefined and t.footnoteRef.oclIsUndefined)
```

7.13.10 StrucDoc.TitleFootnote

7.13.10.1 Description

Specializes StrucDoc.Base

Same functionality as a normal footnote, but the content model in the parts is restricted to the kind of content that can appear in a title.

7.13.10.2 Associations

7.13.10.2.1 parts : StrucDoc.CMTitle[0..* ordered] «Anonymous»: The contents of the footnote.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.10.3 Invariants

- The footnote must contain some content

OCL for Invariants:

```
inv "Some content required": parts->notEmpty
```

7.13.11 StrucDoc.Content

7.13.11.1 Description

Specializes StrucDoc.Base

Used to wrap a string of text so that it can be explicitly referenced, or so that it can suggest rendering characteristics. Content can be nested recursively, which enables wrapping a string of plain text down to as small a chunk as desired.

Content has an optional identifier that can serve as the target of a reference. This identifier, represented as an XML ID attribute, must be unique within the document context. The originalText attribute of a datatype defined in this specification may make explicit reference to the content using the identifier, thereby indicating the original text associated with the datatype.

© ISO 2007 – All rights reserved

7.13.11.2 Attributes

7.13.11.2.1 revised: Revised: can be used to indicate narrative changes from the last version of a CDA document**.** The attribute is limited to a single generation, in that it only reflects the changes from the preceding version of a document. Receivers SHALL interpret the "revised" attribute when rendering by visually distinguishing or suppressing deleted narrative.

Note: If applied, this attribute SHALL be used in conjunction with appropriate document version tracking as defined in the applicable conformance statement for the document context.

If populated, the value of this attribute SHALL be taken from one of these values:

Revised Enumeration			
1	insert	insert	This content was inserted in this revision of the document
1	delete	delete	This content was deleted in this revision of the document

7.13.11.3 Associations

7.13.11.3.1 parts : StrucDoc.CMContent[0..* ordered] «Anonymous»: The contents of the Content.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.11.4 Invariants

- The content must contain some content

OCL for Invariants:

```
inv "Some content required": parts->notEmpty
```

7.13.12 StrucDoc.Caption

7.13.12.1 Description

Specializes StrucDoc.Base

A label for a paragraph, list, list item, table, or table cell. It may also be used within RenderMultiMedia to indicate a label for referenced ObservationMedia and RegionOfInterest entries. A Caption contains plain text and may contain links and footnotes.

Note: If a caption is defined, it SHALL be rendered, and SHALL be presented before any the element with which it is associated.

7.13.12.2 Associations

7.13.12.2.1 parts : StrucDoc.CMInline[0..* ordered] «Anonymous»: The contents of the Content.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

© ISO 2007 – All rights reserved

7.13.12.3 Invariants

- The caption must contain some content

OCL for Invariants:

```
inv "Some content required": parts->notEmpty
```

7.13.13 StrucDoc.Captioned

7.13.13.1 Description

Abstract. Specializes StrucDoc.Base

An abstract ancestor for all types that have captions.

Note: If a caption is defined, it SHALL be rendered, and SHALL be presented before any the element with which it is associated.

7.13.13.2 Attributes

7.13.13.2.1 caption : StrucDoc.Caption : The contents of the Content.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.14 StrucDoc.Paragraph

7.13.14.1 Description

Specializes StrucDoc.Captioned

Similar to the HTML paragraph, which allows blocks of narrative to be broken up into logically consistent structures

7.13.14.2 Associations

7.13.14.2.1 parts: StrucDoc.CMInline[0..* ordered] «Anonymous»: The contents of the Content.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.14.3 Invariants

- The caption must contain some content

OCL for Invariants:

```
inv "Some content required": parts->notEmpty
```


© ISO 2007 – All rights reserved

7.13.15 StrucDoc.CMFootnotes

7.13.15.1 Description

Stereotype: «Choice»

Content model that allows text and footnotes. The choice stereotype denotes that exactly one of the attributes SHALL have a value. All the others must be null.

7.13.15.2 Attributes

7.13.15.2.1 Text : String «XMLText»: Plain text.

Note: Refer to section A.2 for the meaning of the XMLText stereotype.

7.13.15.2.2 footnote : StrucDoc.Footnote: A footnote.

7.13.15.2.3 footnoteRef : StrucDoc.FootnoteRef: A reference to a footnote.

7.13.16 StrucDoc.CMInline

7.13.16.1 Description

Specializes StrucDoc.CMFootnotes

Stereotype: «Choice»

Content model that allows text, footnotes, links, and superscript and subscript text. The choice stereotype denotes that exactly one of the attributes (including inherited attributes) SHALL have a value. All the others must be null.

7.13.16.2 Attributes

7.13.16.2.1 linkHtml : StrucDoc.LinkHtml: A HTML-type link.

7.13.16.2.2 sub : StrucDoc.Sub: Subscript text.

7.13.16.2.3 sup : StrucDoc.Sup: Superscript text.

7.13.17 StrucDoc.CMContent

7.13.17.1 Description

Specializes StrucDoc.CMInline

Content model that allows text, footnotes, links, superscript and subscript text, line breaks, multimedia content, and nested Content items. The choice stereotype denotes that exactly one of the attributes (including inherited attributes) SHALL have a value. All the others must be null.

© ISO 2007 – All rights reserved

7.13.17.2 Attributes

7.13.17.2.1 content : StrucDoc.Content: Nested Content.

7.13.17.2.2 br : StrucDoc.Br: A hard line break.

7.13.17.2.3 renderMultiMedia : StrucDoc.RenderMultiMedia: Multimedia.

7.13.18 StrucDoc.CMGeneral

7.13.18.1 Description

Specializes StrucDoc.CMContent

Content model that allows text, footnotes, links, superscript and subscript text, line breaks, multimedia content, nested Content items, paragraphs, lists, and tables. The choice stereotype denotes that exactly one of the attributes (including inherited attributes) SHALL have a value. All the others must be null.

7.13.18.2 Attributes

7.13.18.2.1 paragraph : StrucDoc.Paragaph: A paragraph of text and other CMContent content.

7.13.18.2.2 list : StrucDoc.List: List based content.

7.13.18.2.3 table : StrucDoc.Table: Table.

7.13.19 StrucDoc.CMTitle

7.13.19.1 Description

Stereotype: «Choice»

Content model that allows text and footnotes, hard line breaks, links, superscript and subscript text, and nested content sections. Multimedia content is not allowed. The choice stereotype denotes that exactly one of the attributes (including inherited attributes) SHALL have a value. All the others must be null.

7.13.19.2 Attributes

7.13.19.2.1 Text : String «XMLText»: Plain text.

Note: Refer to section A.2 for the meaning of the XMLText stereotype.

© ISO 2007 – All rights reserved

7.13.19.2.2 footnote : StrucDoc.Footnote: A footnote.

7.13.19.2.3 footnoteRef : StrucDoc.FootnoteRef: A reference to a footnote.

7.13.19.2.4 br : StrucDoc.Br: A hard line break.

7.13.19.2.5 linkHtml : StrucDoc.LinkHtml: A HTML-type link.

7.13.19.2.6 sub : StrucDoc.Sub: Subscript text.

7.13.19.2.7 sup : StrucDoc.Sup: Superscript text.

7.13.19.2.8 content : StrucDoc.Content: Nested Content.

7.13.20 StrucDoc.List

7.13.20.1 Description

Specializes StrucDoc.Captioned

Similar to an HTML list. There is an optional caption, and one or more items. The list must be ordered or not ordered; this must always be known,

7.13.20.2 Attributes

7.13.20.2.1 listType : StrucDoc.ListType: specifies whether the list is ordered or unordered.

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.ListType Enumeration			
1	ordered	ordered	The list is ordered
1	unordered	unordered	The list is not ordered

The default value is unordered.

Note: Unordered lists are typically rendered with bullets, whereas ordered lists are typically rendered with numbers, although this is not a requirement.

7.13.20.3 Associations

7.13.20.3.1 item : StrucDoc.Item[0..* ordered]: Actual list items.

7.13.20.4 Invariants

- at least one item must be provided

OCL for Invariants:

```
inv "must have at least one item": item->notEmpty
```

© ISO 2007 – All rights reserved

7.13.21 StrucDoc.Item

7.13.21.1 Description

Specializes StrucDoc.Captioned

An item in a list.

7.13.21.2 Associations

7.13.21.2.1 parts : StrucDoc.CMGeneral[0..* ordered] «Anonymous»: The contents of the footnote.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.22 StrucDoc.TableItem

7.13.22.1 Description

Abstract. Specializes StrucDoc.Base

An abstract container for table items that may specify table layout details such as alignment.

Note: Any attributes applied to the table item also apply to any other nested table items unless specifically overridden.

7.13.22.2 Attributes

7.13.22.2.1 align : StrucDoc.Align: The text alignment that applies within the cell.

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.CellAlign Enumeration			
1	left	left	The content is left aligned. This long paragraph serves as an example of left aligned content in a table cell.
1	center	center	The content is center aligned. This long paragraph serves as an example of center aligned content in a table cell.
1	right	right	The content is right aligned. This long paragraph serves as an example of right aligned content in a table cell.
1	justify	justify	The content is justified. This long paragraph serves as an example of content that is justified in a table cell.
1	char	Char	align=char aligns a cell's contents on the character given in the char attribute.

© ISO 2007 – All rights reserved

The default value is left.

7.13.22.2.2 char : String: The character on which to align cells if align is set to char. The default value for the CHAR attribute is the decimal point of the current language--a period in English.

7.13.22.2.3 charoff : StrucDoc.Length: When present, this attribute specifies the offset to the first occurrence of the alignment character on each line. If a line doesn't include the alignment character, it should be horizontally shifted to end at the alignment position. Information Processing Entities SHALL not be required to support this attribute.

7.13.22.2.4 valign : StrucDoc.VAlign: The vertical alignment that applies within the cell.

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.VAlign Enumeration			
1	top	top	The content is aligned with the top of the cell as shown in the caption column.
1	middle	middle	The content is aligned with the bottom of the cell as shown in the caption column.
1	bottom	bottom	The content is aligned with the bottom of the cell as shown in the caption column.
1	baseline	baseline	All cells in the same row as a cell whose valign attribute has this value should have their textual data positioned so that the first text line occurs on a baseline common to all cells in the row. This constraint does not apply to subsequent text lines in these cells

The default value is top.

7.13.23 StrucDoc.TCell

7.13.23.1 Description

Specializes StrucDoc.TableItem

A cell in a table – may be either a normal cell or a header cell

© ISO 2007 – All rights reserved

7.13.23.2 Attributes

7.13.23.2.1 abbr : String: This attribute should be used to provide an abbreviated form of the cell's content, and may be rendered by user agents when appropriate in place of the cell's content. Abbreviated names should be short since user agents may render them repeatedly.

7.13.23.2.2 axis : String: This attribute may be used to place a cell into conceptual categories that can be considered to form axes in an n-dimensional space. User agents may give users access to these categories (e.g., the user may query the user agent for all cells that belong to certain categories, the user agent may present a table in the form of a table of contents, etc.). Please consult the HTML specification for more information. The value of this attribute is a comma-separated list of category names

7.13.23.2.3 headers : Set(String) «XMLIDREF»: This attribute specifies the list of header cells that provide header information for the current data cell. The value of this attribute is a space-separated list of ID references to header cells; those cells must be named by setting their id attribute. Authors generally use the headers attribute to help non-visual user agents render header information about data cells (e.g., header information is spoken prior to the cell data), but the attribute may also be used in conjunction with style sheets. See also the scope attribute.

Refer to section A.2 for the meaning of the XMLIDREF stereotype.

7.13.23.2.4 scope : StrucDoc.CellScope: This attribute specifies the set of data cells for which the current header cell provides header information. This attribute may be used in place of the headers attribute, particularly for simple tables.

This attribute SHALL only be populated for header cells.

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.CellScope Enumeration			
1	Row	row	The current cell provides header information for the rest of the row that contains it
1	Col	col	The current cell provides header information for the rest of the column that contains it.
1	rowgroup	rowgroup	The header cell provides header information for the rest of the row group [p.116] that contains it.
1	colgroup	colgroup	The header cell provides header information for the rest of the column group [p.118] that contains it.

The default value is Col

7.13.23.2.5 rowspan : Integer: The number of rows that this cell spans. Default value is 1

7.13.23.2.6 colspan : Integer: The number of columns that this cell spans. Default value is 1

7.13.23.3 Associations

7.13.23.3.1 parts : StrucDoc.CMGeneral[0..* ordered] «Anonymous»: The contents of the footnote.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

© ISO 2007 – All rights reserved

7.13.23.4 Invariants

- Tables are not allowed to nest directly

OCL for Invariants:

```
inv "no nested tables": parts.forAll(t | t.table.oclIsUndefined)
```

7.13.24 StrucDoc.TRow

7.13.24.1 Description

Specializes StrucDoc.TableItem

A Row in a table

7.13.24.2 Associations

7.13.24.2.1 parts : Sequence(StrucDoc.TRowPart) «Anonymous»: The contents of the row.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.24.3 Invariants

- at least one row part must be provided

OCL for Invariants:

```
inv "must have at least one item": parts->notEmpty
```

7.13.25 StrucDoc.TRowPart

7.13.25.1 Description

Stereotype: «Choice»

Content model that allows cells (td) or header cells (th). The choice stereotype denotes that exactly one of the attributes SHALL have a value. The other must be null.

7.13.25.2 Attributes

7.13.25.2.1 td : StrucDoc.TCell: A cell in a table.

7.13.25.2.2 th : StrucDoc.TCell: A header cell in a table.

Note: Header cells are sometimes rendered differently, such as using bold, and may be repeated after page breaks.

© ISO 2007 – All rights reserved

7.13.26 StrucDoc.TRowGroup

7.13.26.1 Description

Specializes StrucDoc.TableItem

A grop of rows – may be used to associate consistent styling across a group of rows.

7.13.26.2 Attributes

7.13.26.2.1 tr : Sequence(StrucDoc.Trow): The rows in the group.

7.13.26.3 Invariants

- at least one row must be provided

OCL for Invariants:

```
inv "must have at least one item": tr->notEmpty
```

7.13.27 StrucDoc.ColItem

7.13.27.1 Description

Abstract. Specializes StrucDoc.TableItem

Abstract ancestor for common properties of col and colgroup.

7.13.27.2 Attributes

7.13.27.2.1 span : Integer: The number of columns this column definition spans.. Default value is 1.

7.13.27.2.2 width : StrucDoc.Length: The length for the column.

7.13.28 StrucDoc.Col

7.13.28.1 Description

Specializes StrucDoc.ColItem

Applies a consistent style to every cell in a column.

7.13.29 StrucDoc.ColGroup

7.13.29.1 Description

Specializes T+

© ISO 2007 – All rights reserved

Applies a consistent style to every cell in a group of columns.

7.13.29.2 Associations

7.13.29.2.1 col : StrucDoc.Col[0..* ordered]: The columns in this group.

7.13.29.3 Invariants

- at least one column must be provided

OCL for Invariants:

```
inv "must have at least one item": col->notEmpty
```

7.13.30 StrucDoc.Table

7.13.30.1 Description

Specializes StrucDoc.Captioned

A table. May have a caption, and SHALL have at least one row. A table may have optional header and footer rows. All rows are defined in groups. A table may also have col and colgroup elements to define styles for columns.

7.13.30.2 Attributes

7.13.30.2.1 summary : String : This attribute provides a summary of the table's purpose and structure for user agents rendering to non-visual media such as speech and Braille. This is different from the caption in that it must be plain text, and it is usually longer.

7.13.30.2.2 width : StrucDoc.Length : This attribute specifies the desired width of the entire table and is intended for visual user agents.

Note: The rules described in the html specification for table width calculations apply to the tables described here.

7.13.30.2.3 border : StrucDoc.Length : The width of the border.

7.13.30.2.4 frame : StrucDoc.Frame : This attribute specifies which sides of the frame surrounding a table will be visible (i.e. which borders are visible).

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.Frame Enumeration			
1	void	void	No sides
1	above	above	The top side only
1	below	below	The bottom side only
1	hsides	hsides	The right and left sides only

© ISO 2007 – All rights reserved

1	lhs	lhs	The left-hand side only
1	rhs	rhs	The right-hand side only
1	vsides	vsides	The top and bottom sides only
1	box	box	All four sides
1	border	border	All four sides

The default value is void.

7.13.30.2.5 rules : StrucDoc.Rules : This attribute specifies which rules (i.e. borders) will appear between cells within a table. The rendering of rules is user agent dependent.

If populated, the value of this attribute SHALL be taken from one of these values:

StrucDoc.Rules Enumeration			
1	none	none	No rules.
1	groups	groups	Rules will appear between row groups and column only
1	rows	rows	Rules will appear between rows only
1	cols	cols	Rules will appear between columns only
1	all	all	Rules will appear between all rows and columns

The default value is none.

7.13.30.2.6 cellspacing : StrucDoc.Length : This attribute specifies how much space the user agent should leave between the left side of the table and the left-hand side of the leftmost column, the top of the table and the top side of the topmost row, and so on for the right and bottom of the table. The attribute also specifies the amount of space to leave between cells.

7.13.30.2.7 cellpadding : StrucDoc.Length: This attribute specifies the amount of space between the border of the cell and its contents. If the value of this attribute is a pixel length, all four margins should be this distance from the contents. If the value of the attribute is a percentage length, the top and bottom margins should be equally separated from the content based on a percentage of the available vertical space, and the left and right margins should be equally separated from the content based on a percentage of the available horizontal space.

Note: consult the HTML specification for further information.

© ISO 2007 – All rights reserved

7.13.30.3 Associations

7.13.30.3.1 thead : TRowGroup: The optional group of rows that defines the header for the table.

7.13.30.3.2 tfoot : TRowGroup: The optional group of rows that defines the footer for the table.

7.13.30.3.3 tbody : TrowGroup[0..* ordered]: The optional group of rows that defines the body of the table.

7.13.30.4 Invariants

- at least one row must be provided

OCL for Invariants:

```
inv "must have at least one row": thead.tr->count + tfoot.tr->count
      + tbody.tr->count > 0
```

7.13.31 StrucDoc.Text

7.13.31.1 Description

Specializes StrucDoc.Base

A definition of structured text that can be used in healthcare.

The structured text is based on an XHTML-like arrangement that ensures the text is properly marked up with semantics, and provides a common base line for implementation in healthcare.

7.13.31.2 Associations

7.13.31.2.1 parts : StrucDoc.CMGeneral[0..* ordered] «Anonymous»: The contents of the structured text.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

7.13.32 StrucDoc.Title

Specializes StrucDoc.Base

A definition of structured title that can be used in healthcare.

The structured text is based on the Structured Text definition, but only a narrow set of features can be used, consistent with a title rather than a general document.

© ISO 2007 – All rights reserved

7.13.32.1 Associations

7.13.32.1.1 parts : StrucDoc.CMTitle[0..* ordered] «Anonymous»: The contents of the title.

Note: Refer to section A.2 for the meaning of the Anonymous stereotype.

© ISO 2007 – All rights reserved

Annex A: XML Representation

A.1 Introduction

Due to the ubiquitious use of XML as an exchange format, this specification provides a normative XML format for use when instances of these data types are represented in XML. An information processing entity SHALL specify whether this format is adopted or not when claiming conformance to this specification.

The XML representation is created by a simple algorithm described below, and is intended to provide direct support for both schema validation and schema based software development tools. A full schema for the XML representation is provided in an informative appendix below. Although the schema is only informative, an information processing entity that claims conformance with this specification SHALL produce instances that are valid with respect to the schema, though validity with respect to the schema is not sufficient to claim that conformance has been demonstrated.

A.2 Rules for XML Representation

- any valid XML charset can be used as long as as it is consistent with the String Character Set as discussed in section 6.7.4. In particular, ful round-trip encoding between the String Character Set and the XML charset is required.
- xml:lang must be ignored. For content to which language applies, the Data.language or ST.language attribute must be used instead
- All elements must be in some namespace, and the namespace must be defined in the conformance statements of information processing entities that claim conformance with this specification. This specification reserves the namespace "uri:iso.org:21090" for direct applications of these datatypes such as testing environments
- the xml representation (and xml schema) for a type is derived algorithmically from the UML representation
- Each type that specializes ANY is represented by an XML element
- UML Attributes with a stereotype of Binary on their type are represented as an element with text content
    - the format of the text in the element is the format specified for the type base64Binary in W3C schema
- UML Attributes with a stereotype of XML on their type are represented as XML with a single XML element contained
    - the name of the XML element is the name of the UML attribute
    - this is the equivalent of the schema type anyType
- UML Attributes with a stereotype of XMLID are represented as an XML ID atttribute. The name of the attribute is the same is the UML attribute.

© ISO 2007 – All rights reserved

- UML Attributes with a stereotype of XMLIDREF are represented as an XML IDREF atttribute. The name of the attribute is the same is the UML attribute. If the type of the UML attribute is a UML collection, then the attribute will be a space delimited list of XML IDREF tokens.

- UML Attributes with a stereotype of XMLText are represented as XML text content. The may only be one UML attribute with this stereotype on a class.

- UML Attributes with a type derived from Classifier are represented as an element

    - the name of the XML element is the name of the UML attribute

    - the format of the element is that specified for the type following these rules recursively

- UML attributes with a type derived from a UML primitive class are represented as an attribute

    - the name of the XML attribute is the name of the UML attribute

    - the attribute has no namespace

    - the content of of the attribute must conform to the W3C schema type that matches the UML type as specified in the table below

    - If the value of the attribute is equal to it's default value, the attribute does not need to be represented in the XML. The default value for attributes is null unless specified otherwise in the UML diagrams

- UML Attributes with a type derived from a Collection of UML Classifiers are represented by a sequence of XML elements

    - the name of the elements is the name of the attribute

    - there is one element for each item in the collection

    - the format of the element is that specified for the type following these rules recursively

- UML Attributes with a type derived from a Collection of UML primitives are represented by an XML attribute

    - the name of the attribute is the name of the attribute

    - the attribute has no namespace

    - the content of the attribute must be a space separated list of tokens that conform to the schema type that matches the UML type as specified in the table below

- UML Associations with a stereotype of "Anonymous" are not represented by any element at all

- the xsi:null element may be used to represent null objects as appropriate. If a UML attribute is not represented in the XML, this is identical in meaning to providing an element with a xsi:null attribute

- If the type being represented has a type other than that specified in the UML (i.e. a specialization), then the xsi:type attribute of the element must be provided. xsi:type may be used at any time

© ISO 2007 – All rights reserved

- The schema type name of a type is the same as it's UML name, except for bound parameterised classes
- For bound parameterized classes (where the class has type parameters) then the actual parameters bound to the class are added to the type using _ as a separator. i.e. SET(AD) becomes SET_AD
- If an attribute is contained by reference, then this is represented using XML ID/IDREF attributes. The reference must be contained within same document. The attribute itself may make other constraints about where the target of the reference might be found. The reference may be a forward reference.

A.3 UML / XML schema type mapping

This table lists the mapping from UML primitive types to XML schema types:

UML primitive type	XML Schema Type
Boolean	boolean
Integer	integer
Real	double
String	string

If the UML type has a stereotype Uri, then the W3C schema type anyURI is used instead of string.

A.4 XML Schema

An XML schema can be derived based on the mapping rules detailed above. A copy of one such schema is found in Appendix D. This schema can be used to validate whether XML instances conform to the some of the rules laid out in this specification. The schema includes schematron statements that represent some of the OCL constraints in this specification.

The schema cannot provide full validation. The fact that any particular instance passes validation against this schema using a schema validation tool does not prove that the instance is conformant with regards to this specification. Applications will have to undergo further data validation to check data intregrity rules that are not represented in the schema before conformance can be assumed. If an instance fails to pass validation using a schema tool, and it is not due to a bug or missing feature in the schema validation tool, then the instance is not conformant to this specification.

© ISO 2007 – All rights reserved

Annex B: UML support types

B.1 Introduction

This section declares some UML types that are needed to support the UML type declarations for the UML types above.

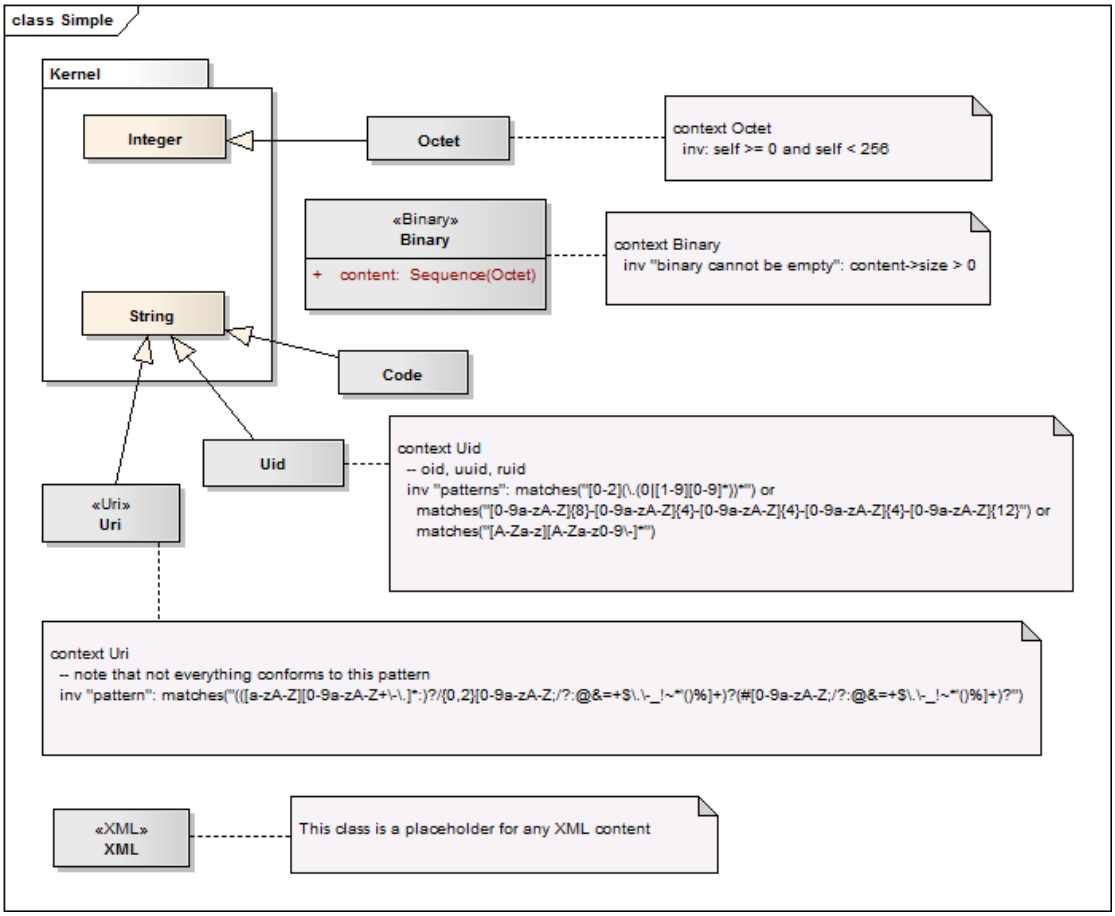


Figure B1: UML Support Data types

B.2 UML data types

B.2.1 Octet

Represents a number that can take values from 0-255 (also sometimes known as a byte). This is defined to allow a formal UML definition of Binary.

B.2.2 Binary

Represents a sequence of Octets. This type is specifically created to allow UML declarations to refer to binary content (also sometimes known as streams).

© ISO 2007 – All rights reserved

This type as a stereotype <<Binary>> to assist with automated mappings to specific platforms, which generally provide specific types for dealing with binary content.

B.2.3 Code

A simple string that has a restricted set of possible values. Code is generally used where the code list is restricted by some external standard, such as an RFC.

B.2.4 Uid

A string that is an identifier, and therefore must be either a string representation of an OID (see ISO/IEC 8824:1990(E)), a UUID (see Open Group, CDE 1.1 Remote Procedure Call specification, Appendix A), or a simple token taken from a list of controlled names for the context in which the Uid is used.

Notes:

- UUIDs SHALL always be represented in uppercase
- Uid comparison is always case sensitive

B.2.5 Uri

A simple internet URL or URI. This is an extension of string but has a stereotype <<URI>> to assist with automated mappings to specific platforms, which may provide a specific type for dealing with binary content.

B.2.6 XML

A placeholder for any XML content. This is used in the Data class to allow for either binary content or an XML object model.

© ISO 2007 – All rights reserved

Annex C: RM-ODP Viewpoint Mappings

The Reference Model of Open Distributed Processing (RM-ODP) is a joint effort by ISO/IEC and ITU-T which provides a co-ordinating framework for the standardisation of open distributed processing (ODP). The RM-ODP family of recommendations and international standards defines essential concepts necessary to specify open distributed processing systems from five prescribed viewpoints and provides a well-developed framework for the structuring of specifications for large-scale, distributed systems.

RM-ODP defines five viewpoints for describing a system.

- the **enterprise viewpoint**, which is concerned with the purpose, scope and policies governing the activities of the specified system within the organization of which it is a part;
- the **information viewpoint**, which is concerned with the kinds of information handled by the system and constraints on the use and interpretation of that information;
- the **computational viewpoin**t, which is concerned with the functional decomposition of the system into a set of objects that interact at interfaces - enabling system distribution;
- the **engineering viewpoint**, which is concerned with the infrastructure required to support system distribution;
- the **technology viewpoint**, which is concerned with the choice of technology to support system distribution.

Though there is not a direct mapping between any of these five viewpoints and this datatype specification, there are conceptual similarities between these views and the data type specifications.

The HL7 V3 Abstract Data Types specification is conceptually similar to the information viewpoint: it is concerned with the kinds of information that may be handled, and constraints on the use and interpretation of the information. It is not directly concerned with computation questions that arise when systems are actually designed.

This specification is conceptually similar to the computational view point: it is concerned with how to represent the informational view described in the The HL7 V3 Abstract Data Types in a set of objects, and to deal with computational issues such as how null objects and null flavors interact with each other.

This specification allows for tools to take the UML specifications and create genuine technology bound implementations, such as generated code in a particular computing language. These generated specifications relate to the technology and engineering view points and their interface with the computational viewpoint.

© ISO 2007 – All rights reserved

Annex D: HL7 V3 Abstract Datatypes Mapping

The HL7 V3 Abstract Data Types Specification describes the data types used by HL7 V3 in an purely semantic fashion. The data types are derived in a fashion that is completely independent of any other specification, in much the same fashion as ISO 11404 defines the General Purpose Data Types.

This specification is an implementation of the HL7 V3 Abstract Data Types. This UML diagram summarises the types defined in the HL7 V3 Abstract Data Types Specification

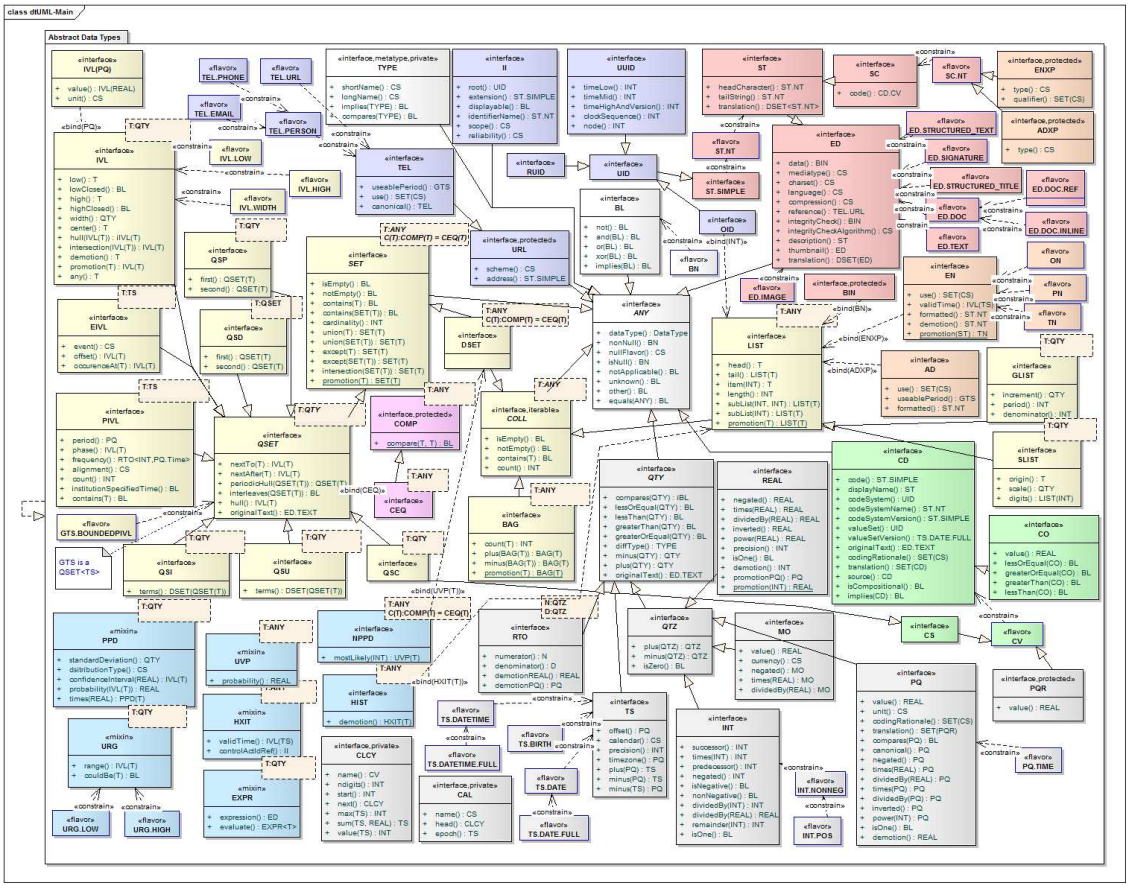


Figure C1: Abstract Datatypes Summary

Note: In this diagram, the HL7 Abstract Data Types are defined as interfaces. There is a custom stereotype used called “mixin”. A mixin interface acts like a parameterised interface, but rather than expressing properties of the type of the parameter, the interface itself extends the interface that it takes as a parameter. This is a technique that is not supported in many implementation technologies, and is also not directly supported in UML.

The following UML diagrams describe how the ISO data type classes implement the HL7 V3 Abstract Data Types

© ISO 2001 – All rights reserved

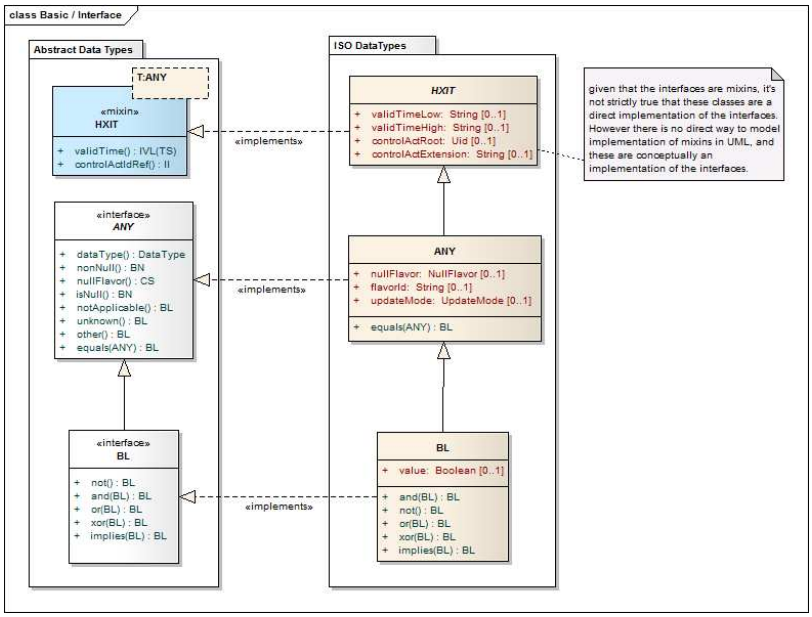


Figure C2: Basic Type Mappings

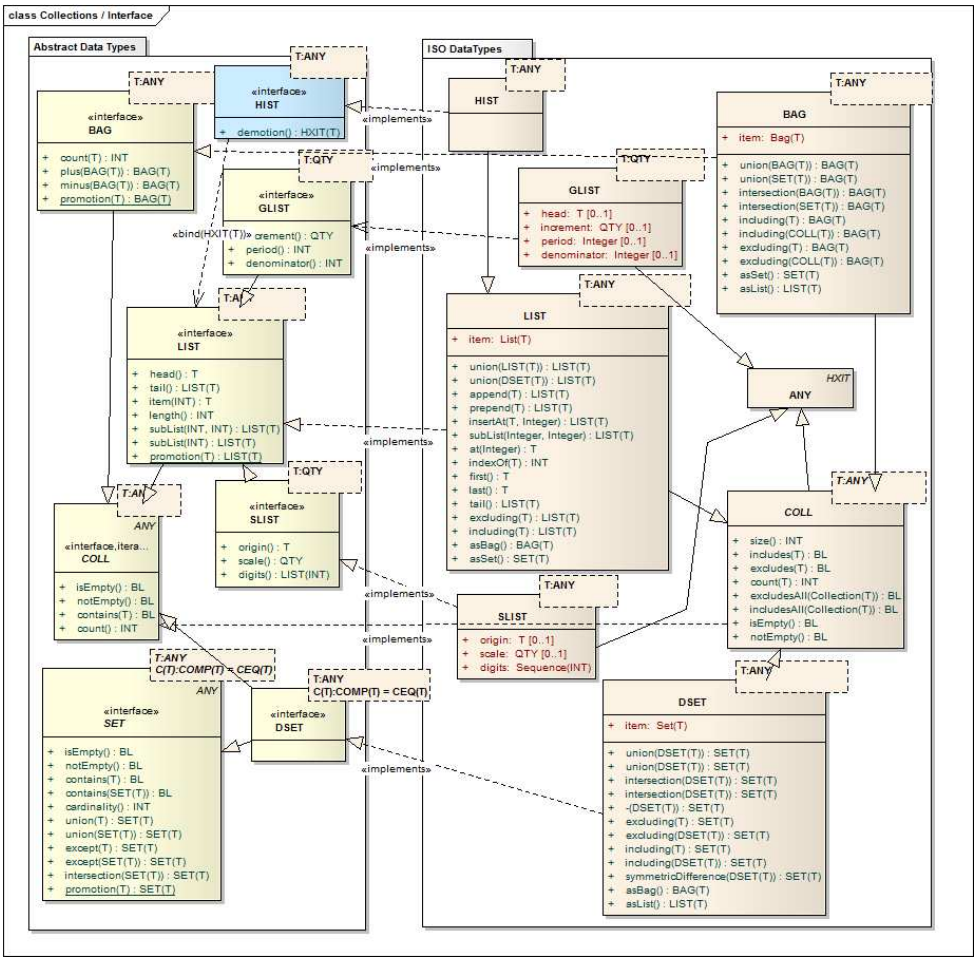


Figure C3: Collection Type Mappings

© ISO 2007 – All rights reserved

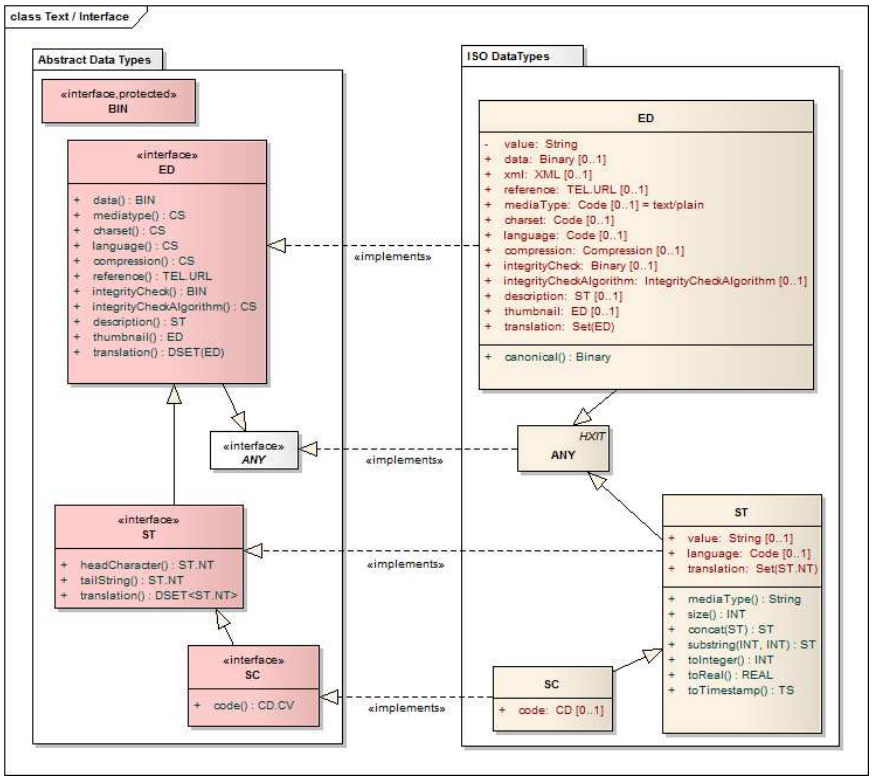


Figure C4: Text Type Mappings

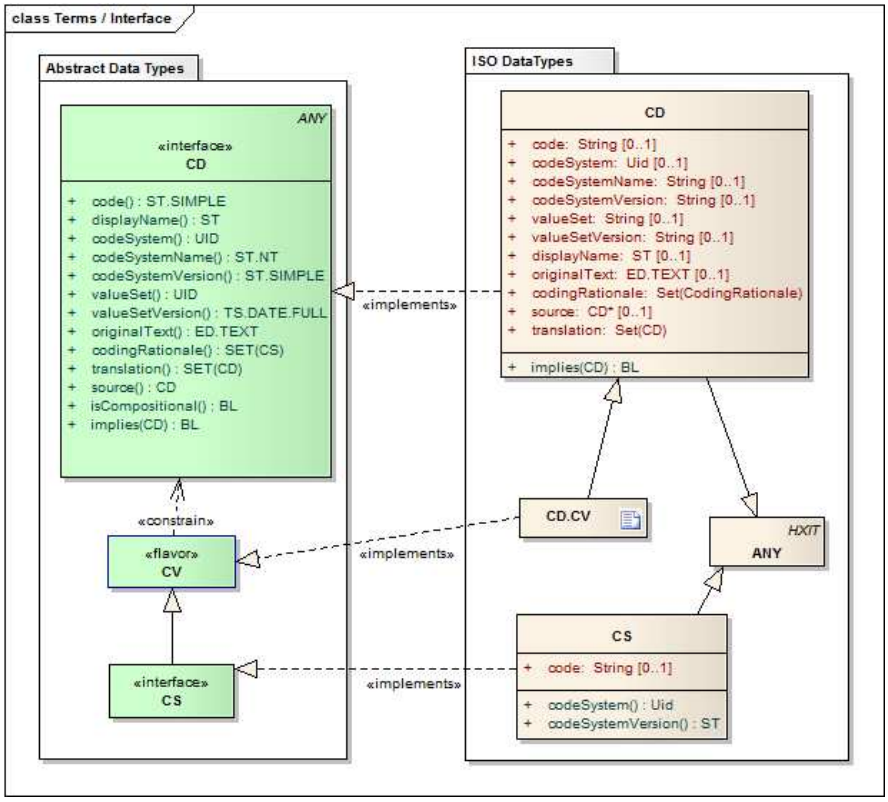


Figure C5: Text Type Mappings

© ISO 2007 – All rights reserved

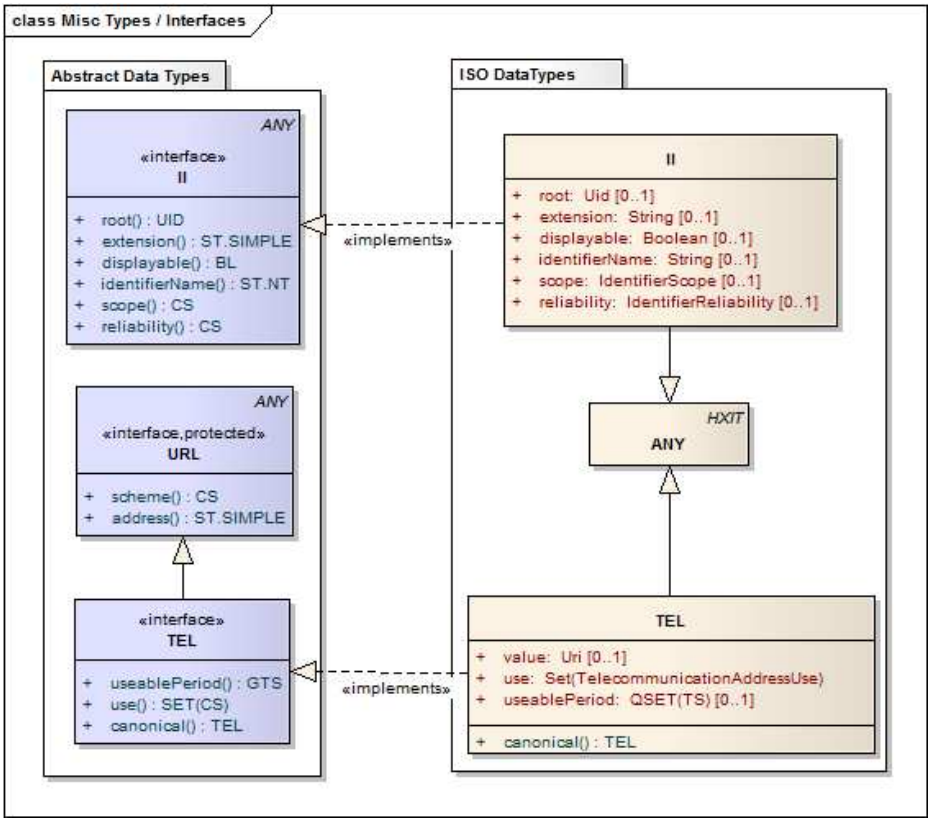


Figure C6: Identification and Location Type Mappings

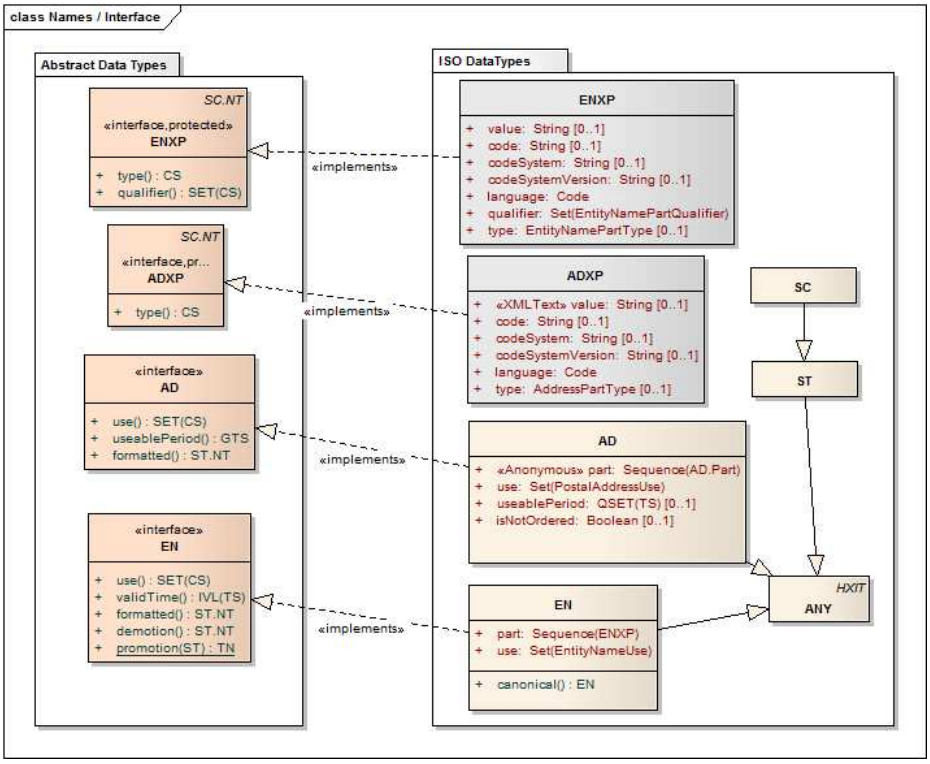


Figure C7: Name and Address Type Mappings

© ISO 2007 – All rights reserved

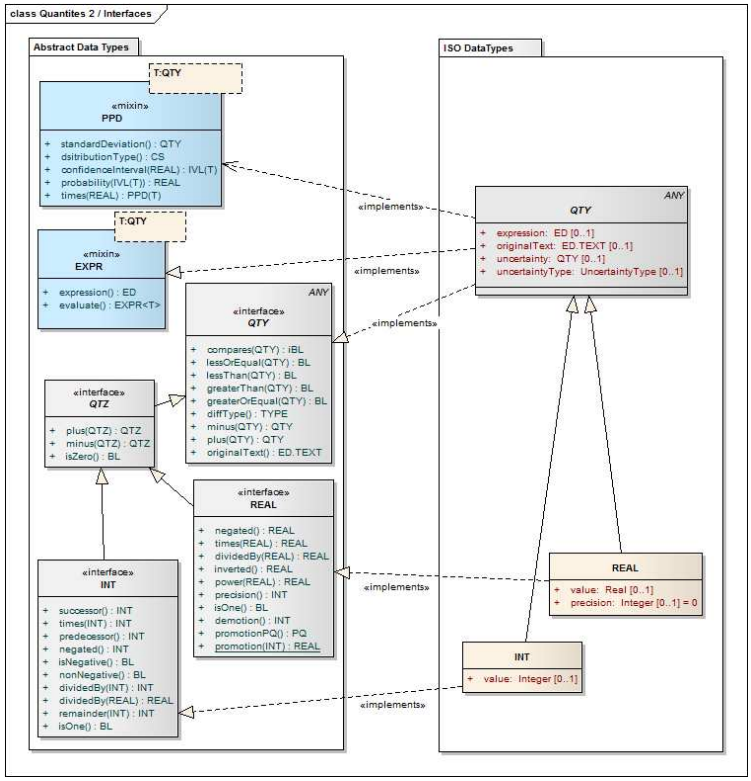


Figure C8: Basic Quantity Type Mappings

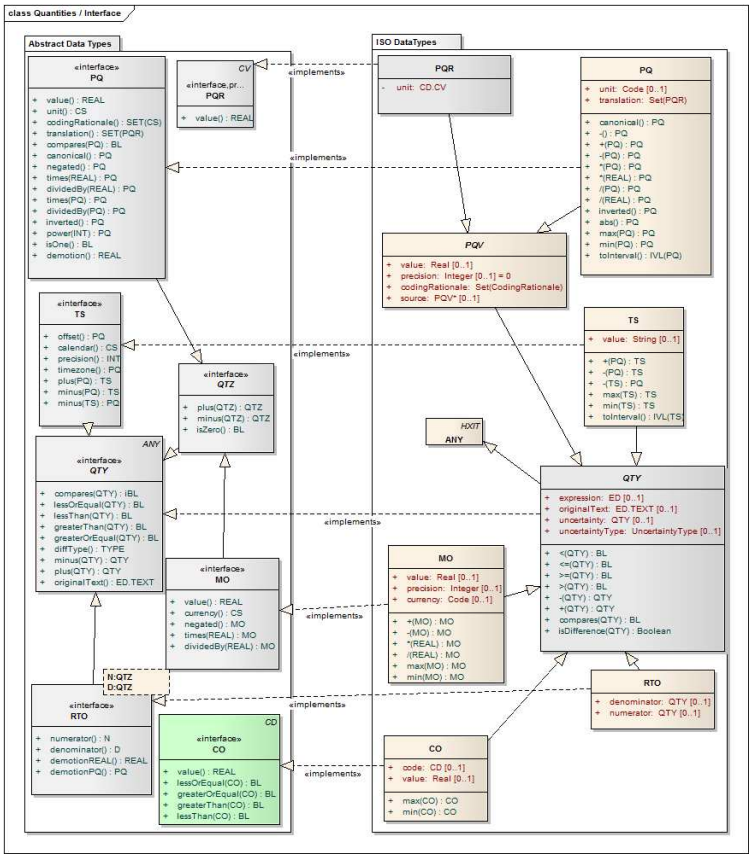


Figure C9: Quantity Type Mappings

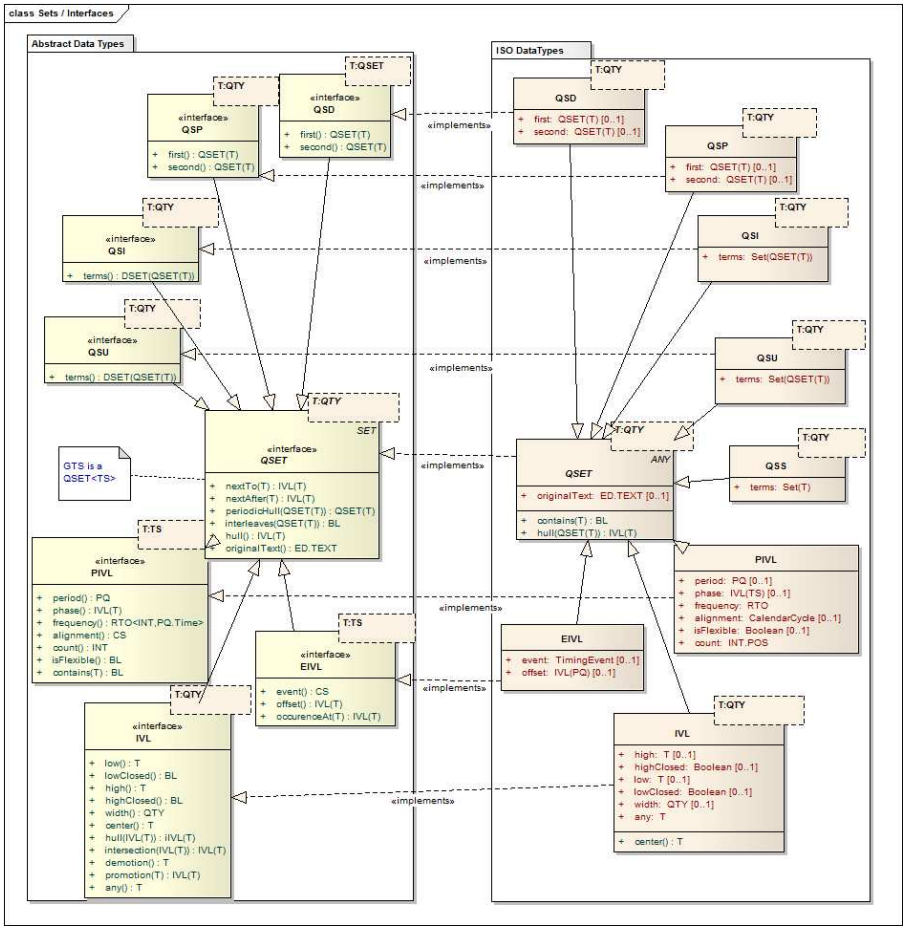


Figure C10: Quantity Set Type Mappings

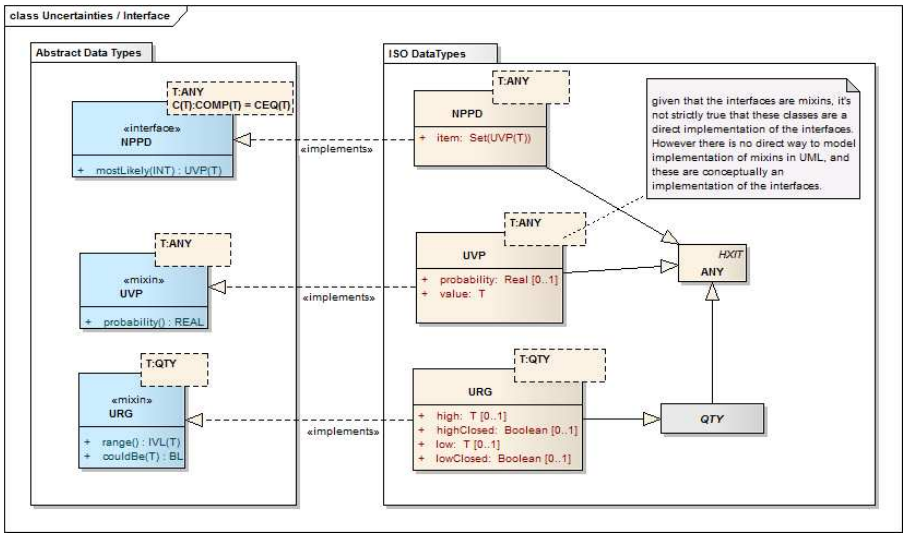


Figure C11: Uncertainty Set Type Mappings

© ISO 2007 – All rights reserved

Annex E: Informative schema for XML representation

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://www.ascc.net/xml/schematron">
  <xsd:simpleType name="NullFlavor">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NI" />
      <xsd:enumeration value="INV" />
      <xsd:enumeration value="OTH" />
      <xsd:enumeration value="NINF" />
      <xsd:enumeration value="PINF" />
      <xsd:enumeration value="UNC" />
      <xsd:enumeration value="DER" />
      <xsd:enumeration value="UNK" />
      <xsd:enumeration value="ASKU" />
      <xsd:enumeration value="NAV" />
      <xsd:enumeration value="QS" />
      <xsd:enumeration value="NASK" />
      <xsd:enumeration value="TRC" />
      <xsd:enumeration value="MSK" />
      <xsd:enumeration value="NA" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="UpdateMode">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="A" />
      <xsd:enumeration value="AU" />
      <xsd:enumeration value="U" />
      <xsd:enumeration value="R" />
      <xsd:enumeration value="I" />
      <xsd:enumeration value="D" />
      <xsd:enumeration value="K" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="Compression">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="DF" />
      <xsd:enumeration value="GZ" />
      <xsd:enumeration value="ZL" />
      <xsd:enumeration value="Z" />
      <xsd:enumeration value="BZ" />
      <xsd:enumeration value="Z7" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="IntegrityCheckAlgorithm">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="SHA1" />
      <xsd:enumeration value="SHA256" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="TelecommunicationAddressUse">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="H" />
      <xsd:enumeration value="HP" />
      <xsd:enumeration value="HV" />
      <xsd:enumeration value="WP" />
      <xsd:enumeration value="DIR" />
      <xsd:enumeration value="PUB" />
      <xsd:enumeration value="BAD" />
      <xsd:enumeration value="TMP" />
      <xsd:enumeration value="AS" />
      <xsd:enumeration value="EC" />
      <xsd:enumeration value="MC" />
      <xsd:enumeration value="PG" />
    </xsd:restriction>
  </xsd:simpleType>
```

© ISO 2001 – All rights reserved

```
<xsd:simpleType name="set_TelecommunicationAddressUse">
  <xsd:list itemType="TelecommunicationAddressUse" />
</xsd:simpleType>
<xsd:simpleType name="IdentifierScope">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="BUSN" />
    <xsd:enumeration value="OBJ" />
    <xsd:enumeration value="VER" />
    <xsd:enumeration value="VW" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="IdentifierReliability">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="ISS" />
    <xsd:enumeration value="VER" />
    <xsd:enumeration value="UNV" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="AddressPartType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AL" />
    <xsd:enumeration value="ADL" />
    <xsd:enumeration value="UNID" />
    <xsd:enumeration value="UNIT" />
    <xsd:enumeration value="DAL" />
    <xsd:enumeration value="DINST" />
    <xsd:enumeration value="DINSTA" />
    <xsd:enumeration value="DINSTQ" />
    <xsd:enumeration value="DMOD" />
    <xsd:enumeration value="DMODID" />
    <xsd:enumeration value="SAL" />
    <xsd:enumeration value="BNR" />
    <xsd:enumeration value="BNN" />
    <xsd:enumeration value="BNS" />
    <xsd:enumeration value="STR" />
    <xsd:enumeration value="STB" />
    <xsd:enumeration value="STTYP" />
    <xsd:enumeration value="DIR" />
    <xsd:enumeration value="INT" />
    <xsd:enumeration value="CAR" />
    <xsd:enumeration value="CEN" />
    <xsd:enumeration value="CNT" />
    <xsd:enumeration value="CPA" />
    <xsd:enumeration value="CTY" />
    <xsd:enumeration value="DEL" />
    <xsd:enumeration value="POB" />
    <xsd:enumeration value="PRE" />
    <xsd:enumeration value="STA" />
    <xsd:enumeration value="ZIP" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="PostalAddressUse">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="H" />
    <xsd:enumeration value="HP" />
    <xsd:enumeration value="HV" />
    <xsd:enumeration value="WP" />
    <xsd:enumeration value="DIR" />
    <xsd:enumeration value="PUB" />
    <xsd:enumeration value="BAD" />
    <xsd:enumeration value="TMP" />
    <xsd:enumeration value="PHYS" />
    <xsd:enumeration value="PST" />
    <xsd:enumeration value="ABC" />
    <xsd:enumeration value="IDE" />
    <xsd:enumeration value="SYL" />
    <xsd:enumeration value="SRCH" />
    <xsd:enumeration value="SNDX" />
    <xsd:enumeration value="PHON" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="set_PostalAddressUse">
```


© ISO 2007 – All rights reserved

```
  <xsd:list itemType="PostalAddressUse" />
</xsd:simpleType>
<xsd:simpleType name="EntityNamePartType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="FAM" />
    <xsd:enumeration value="GIV" />
    <xsd:enumeration value="PFX" />
    <xsd:enumeration value="SFX" />
    <xsd:enumeration value="DEL" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="EntityNamePartQualifier">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="LS" />
    <xsd:enumeration value="AC" />
    <xsd:enumeration value="NB" />
    <xsd:enumeration value="PR" />
    <xsd:enumeration value="VV" />
    <xsd:enumeration value="AD" />
    <xsd:enumeration value="BR" />
    <xsd:enumeration value="SP" />
    <xsd:enumeration value="CL" />
    <xsd:enumeration value="IN" />
    <xsd:enumeration value="TITLE" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="set_EntityNamePartQualifier">
  <xsd:list itemType="EntityNamePartQualifier" />
</xsd:simpleType>
<xsd:simpleType name="EntityNameUse">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="C" />
    <xsd:enumeration value="I" />
    <xsd:enumeration value="L" />
    <xsd:enumeration value="P" />
    <xsd:enumeration value="A" />
    <xsd:enumeration value="R" />
    <xsd:enumeration value="OR" />
    <xsd:enumeration value="SRCH" />
    <xsd:enumeration value="PHON" />
    <xsd:enumeration value="SNDX" />
    <xsd:enumeration value="ABC" />
    <xsd:enumeration value="SYL" />
    <xsd:enumeration value="IDE" />
    <xsd:enumeration value="ASGN" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="set_EntityNameUse">
  <xsd:list itemType="EntityNameUse" />
</xsd:simpleType>
<xsd:simpleType name="CalendarCycle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="CY" />
    <xsd:enumeration value="MY" />
    <xsd:enumeration value="CM" />
    <xsd:enumeration value="CW" />
    <xsd:enumeration value="WY" />
    <xsd:enumeration value="DM" />
    <xsd:enumeration value="CD" />
    <xsd:enumeration value="DY" />
    <xsd:enumeration value="DW" />
    <xsd:enumeration value="HD" />
    <xsd:enumeration value="CH" />
    <xsd:enumeration value="NH" />
    <xsd:enumeration value="CN" />
    <xsd:enumeration value="SN" />
    <xsd:enumeration value="CS" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="TimingEvent">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="HS" />
```

© ISO 2007 – All rights reserved

```
      <xsd:enumeration value="AC" />
      <xsd:enumeration value="ACM" />
      <xsd:enumeration value="ACD" />
      <xsd:enumeration value="ACV" />
      <xsd:enumeration value="IC" />
      <xsd:enumeration value="ICM" />
      <xsd:enumeration value="ICD" />
      <xsd:enumeration value="ICV" />
      <xsd:enumeration value="PC" />
      <xsd:enumeration value="PCM" />
      <xsd:enumeration value="PCD" />
      <xsd:enumeration value="PCV" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="CodingRationale">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="O" />
      <xsd:enumeration value="P" />
      <xsd:enumeration value="R" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="set_CodingRationale">
    <xsd:list itemType="CodingRationale" />
  </xsd:simpleType>
  <xsd:simpleType name="UncertaintyType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="U" />
      <xsd:enumeration value="N" />
      <xsd:enumeration value="LN" />
      <xsd:enumeration value="G" />
      <xsd:enumeration value="E" />
      <xsd:enumeration value="X2" />
      <xsd:enumeration value="T" />
      <xsd:enumeration value="F" />
      <xsd:enumeration value="B" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType mixed="true" name="StrucDoc.Text">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="footnote" type="StrucDoc.Footnote" />
        <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
        <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
        <xsd:element name="sub" type="StrucDoc.Sub" />
        <xsd:element name="sup" type="StrucDoc.Sup" />
        <xsd:element name="content" type="StrucDoc.Content" />
        <xsd:element name="br" type="StrucDoc.Br" />
        <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
        <xsd:element name="paragraph" type="StrucDoc.Paragraph" />
        <xsd:element name="list" type="StrucDoc.List" />
        <xsd:element name="table" type="StrucDoc.Table" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:attributeGroup name="StrucDoc.Base">
    <xsd:attribute name="ID" type="xsd:ID" use="optional" />
    <xsd:attribute name="language" type="Code" use="optional" />
    <xsd:attribute name="styleCode" type="set_Code" use="optional" />
  </xsd:attributeGroup>
  <xsd:simpleType name="XmlID">
    <xsd:restriction base="xsd:ID" />
  </xsd:simpleType>
  <xsd:simpleType name="Code">
    <xsd:restriction base="xsd:string" />
  </xsd:simpleType>
  <xsd:simpleType name="set_Code">
    <xsd:list itemType="Code" />
  </xsd:simpleType>
  <xsd:complexType mixed="true" name="StrucDoc.Footnote">
    <xsd:annotation>
      <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
      <sch:pattern name="Some content required">
        <sch:rule abstract="true" id="StrucDoc.Footnote-0">
          <sch:assert test="count(*|text()[normalize-space(.)!=&#39;&#39;])!=0" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="footnote" type="StrucDoc.Footnote" />
      <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
      <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
      <xsd:element name="sub" type="StrucDoc.Sub" />
      <xsd:element name="sup" type="StrucDoc.Sup" />
      <xsd:element name="content" type="StrucDoc.Content" />
      <xsd:element name="br" type="StrucDoc.Br" />
      <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
      <xsd:element name="paragraph" type="StrucDoc.Paragraph" />
      <xsd:element name="list" type="StrucDoc.List" />
      <xsd:element name="table" type="StrucDoc.Table" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:attributeGroup ref="StrucDoc.Base" />
</xsd:complexType>
<xsd:complexType name="StrucDoc.FootnoteRef">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="must have a reference">
        <sch:rule abstract="true" id="StrucDoc.FootnoteRef-0">
          <sch:assert test="@IDREF" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:attribute name="IDREF" type="xsd:IDREF" use="optional" />
  <xsd:attributeGroup ref="StrucDoc.Base" />
</xsd:complexType>
<xsd:simpleType name="XmlIDREF">
  <xsd:restriction base="xsd:IDREF" />
</xsd:simpleType>
<xsd:complexType mixed="true" name="StrucDoc.LinkHtml">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="must have at least one item">
        <sch:rule abstract="true" id="StrucDoc.LinkHtml-0">
          <sch:assert test="count(*|text()[normalize-space(.)!=&#39;&#39;])!=0" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="footnote" type="StrucDoc.Footnote" />
      <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" />
  <xsd:attribute name="href" type="xsd:string" use="optional" />
  <xsd:attribute name="rel" type="xsd:string" use="optional" />
  <xsd:attribute name="rev" type="xsd:string" use="optional" />
  <xsd:attribute name="title" type="xsd:string" use="optional" />
  <xsd:attributeGroup ref="StrucDoc.Base" />
</xsd:complexType>
<xsd:complexType mixed="true" name="StrucDoc.Sub">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="text must not be empty">
        <sch:rule abstract="true" id="StrucDoc.Sub-0">
          <sch:assert test="string-length(text()) != 0" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
    </xsd:annotation>
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Sup">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="text must not be empty">
          <sch:rule abstract="true" id="StrucDoc.Sup-0">
            <sch:assert test="string-length(text()) != 0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Content">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="footnote" type="StrucDoc.Footnote" />
        <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
        <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
        <xsd:element name="sub" type="StrucDoc.Sub" />
        <xsd:element name="sup" type="StrucDoc.Sup" />
        <xsd:element name="content" type="StrucDoc.Content" />
        <xsd:element name="br" type="StrucDoc.Br" />
        <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="revised" type="StrucDoc.Revised" use="optional" />
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:simpleType name="StrucDoc.Revised">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="insert" />
      <xsd:enumeration value="delete" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="StrucDoc.Br" />
  <xsd:complexType name="StrucDoc.RenderMultiMedia">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one reference">
          <sch:rule abstract="true" id="StrucDoc.RenderMultiMedia-0">
            <sch:assert test="@referencedObject" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="caption" type="StrucDoc.Caption" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="referencedObject" type="set_IDREF" use="optional" />
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Caption">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="footnote" type="StrucDoc.Footnote" />
        <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
        <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
        <xsd:element name="sub" type="StrucDoc.Sub" />
        <xsd:element name="sup" type="StrucDoc.Sup" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:simpleType name="set_IDREF">
    <xsd:list itemType="xsd:IDREF" />
  </xsd:simpleType>
  <xsd:complexType mixed="true" name="StrucDoc.Paragraph">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.Captioned">
        <xsd:sequence>
          <xsd:choice minOccurs="0" maxOccurs="unbounded">
```


© ISO 2007 – All rights reserved

```
            <xsd:element name="footnote" type="StrucDoc.Footnote" />
            <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
            <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
            <xsd:element name="sub" type="StrucDoc.Sub" />
            <xsd:element name="sup" type="StrucDoc.Sup" />
            <xsd:element name="content" type="StrucDoc.Content" />
            <xsd:element name="br" type="StrucDoc.Br" />
            <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Captioned">
    <xsd:sequence>
      <xsd:element name="caption" type="StrucDoc.Caption" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.List">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.Captioned">
        <xsd:sequence>
          <xsd:element name="item" type="StrucDoc.Item" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="listType" type="StrucDoc.ListType" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:simpleType name="StrucDoc.ListType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ordered_" />
      <xsd:enumeration value="unordered_" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType mixed="true" name="StrucDoc.Item">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.Captioned">
        <xsd:sequence>
          <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="footnote" type="StrucDoc.Footnote" />
            <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
            <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
            <xsd:element name="sub" type="StrucDoc.Sub" />
            <xsd:element name="sup" type="StrucDoc.Sup" />
            <xsd:element name="content" type="StrucDoc.Content" />
            <xsd:element name="br" type="StrucDoc.Br" />
            <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
            <xsd:element name="paragraph" type="StrucDoc.Paragraph" />
            <xsd:element name="list" type="StrucDoc.List" />
            <xsd:element name="table" type="StrucDoc.Table" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Table">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="columns">
          <sch:rule abstract="true" id="StrucDoc.Table-0">
            <sch:assert test="not(col) or not(column)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.Captioned">
        <xsd:sequence>
          <xsd:element name="col" type="StrucDoc.Col" minOccurs="0" maxOccurs="unbounded" />
          <xsd:element name="colgroup" type="StrucDoc.ColGroup" minOccurs="0"
maxOccurs="unbounded" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="thead" type="StrucDoc.TRowGroup" minOccurs="0" maxOccurs="1" />
          <xsd:element name="tfoot" type="StrucDoc.TRowGroup" minOccurs="0" maxOccurs="1" />
          <xsd:element name="tbody" type="StrucDoc.TRowGroup" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="summary" type="xsd:string" use="optional" />
        <xsd:attribute name="width" type="StrucDoc.Length" use="optional" />
        <xsd:attribute name="border" type="StrucDoc.Length" use="optional" />
        <xsd:attribute name="frame" type="StrucDoc.Frame" use="optional" />
        <xsd:attribute name="rules" type="StrucDoc.Rules" use="optional" />
        <xsd:attribute name="cellspacing" type="StrucDoc.Length" use="optional" />
        <xsd:attribute name="cellpadding" type="StrucDoc.Length" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="StrucDoc.Col">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.ColItem" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="StrucDoc.ColItem">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.TableItem">
        <xsd:attribute name="span" type="xsd:int" default="1" use="optional" />
        <xsd:attribute name="width" type="StrucDoc.Length" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="StrucDoc.TableItem">
    <xsd:attribute name="align" type="StrucDoc.Align" use="optional" />
    <xsd:attribute name="char" type="xsd:string" use="optional" />
    <xsd:attribute name="charoff" type="StrucDoc.Length" use="optional" />
    <xsd:attribute name="valign" type="StrucDoc.VAlign" use="optional" />
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:simpleType name="StrucDoc.Align">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="left" />
      <xsd:enumeration value="center" />
      <xsd:enumeration value="right" />
      <xsd:enumeration value="justify" />
      <xsd:enumeration value="char" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="StrucDoc.Length">
    <xsd:restriction base="xsd:string" />
  </xsd:simpleType>
  <xsd:simpleType name="StrucDoc.VAlign">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="top" />
      <xsd:enumeration value="middle" />
      <xsd:enumeration value="bottom" />
      <xsd:enumeration value="baseline" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="StrucDoc.ColGroup">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.ColItem">
        <xsd:sequence>
          <xsd:element name="col" type="StrucDoc.Col" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="StrucDoc.TRowGroup">
    <xsd:complexContent>
      <xsd:extension base="StrucDoc.TableItem">
        <xsd:sequence>
          <xsd:element name="tr" type="StrucDoc.TRow" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
</xsd:complexType>
<xsd:complexType name="StrucDoc.TRow">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="cells">
        <sch:rule abstract="true" id="StrucDoc.TRow-0">
          <sch:assert test="th|td" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="StrucDoc.TableItem">
      <xsd:sequence>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="th" type="StrucDoc.TCell" />
          <xsd:element name="td" type="StrucDoc.TCell" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType mixed="true" name="StrucDoc.TCell">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="no nested tables">
        <sch:rule abstract="true" id="StrucDoc.TCell-0">
          <sch:assert test="not(@table)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="StrucDoc.TableItem">
      <xsd:sequence>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="footnote" type="StrucDoc.Footnote" />
          <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
          <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
          <xsd:element name="sub" type="StrucDoc.Sub" />
          <xsd:element name="sup" type="StrucDoc.Sup" />
          <xsd:element name="content" type="StrucDoc.Content" />
          <xsd:element name="br" type="StrucDoc.Br" />
          <xsd:element name="renderMultiMedia" type="StrucDoc.RenderMultiMedia" />
          <xsd:element name="paragraph" type="StrucDoc.Paragraph" />
          <xsd:element name="list" type="StrucDoc.List" />
          <xsd:element name="table" type="StrucDoc.Table" />
        </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="abbr" type="xsd:string" use="optional" />
      <xsd:attribute name="axis" type="xsd:string" use="optional" />
      <xsd:attribute name="headers" type="set_IDREF" use="optional" />
      <xsd:attribute name="scope" type="StrucDoc.CellScope" use="optional" />
      <xsd:attribute name="rowspan" type="xsd:int" default="1" use="optional" />
      <xsd:attribute name="colspan" type="xsd:int" default="1" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="StrucDoc.CellScope">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="row" />
    <xsd:enumeration value="col" />
    <xsd:enumeration value="rowgroup" />
    <xsd:enumeration value="colgroup" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StrucDoc.Frame">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="void" />
    <xsd:enumeration value="above" />
    <xsd:enumeration value="below" />
    <xsd:enumeration value="hsides" />
```

© ISO 2007 – All rights reserved

```
      <xsd:enumeration value="lhs" />
      <xsd:enumeration value="rhs" />
      <xsd:enumeration value="vsides" />
      <xsd:enumeration value="box" />
      <xsd:enumeration value="border" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="StrucDoc.Rules">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="none" />
      <xsd:enumeration value="groups" />
      <xsd:enumeration value="rows" />
      <xsd:enumeration value="cols" />
      <xsd:enumeration value="all" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="Uid">
    <xsd:restriction base="xsd:string" />
  </xsd:simpleType>
  <xsd:simpleType name="Uri">
    <xsd:restriction base="xsd:anyURI" />
  </xsd:simpleType>
  <xsd:complexType name="BL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value if not null">
          <sch:rule abstract="true" id="BL-0">
            <sch:assert test="@nullFlavor or (not(@nullFlavor) and @value)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:attribute name="value" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ANY">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no ANY unless null">
          <sch:rule abstract="true" id="ANY-0">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="HXIT">
        <xsd:attribute name="nullFlavor" type="NullFlavor" use="optional" />
        <xsd:attribute name="flavorId" type="xsd:string" use="optional" />
        <xsd:attribute name="updateMode" type="UpdateMode" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="HXIT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="extension requires root">
          <sch:rule abstract="true" id="HXIT-0">
            <sch:assert test="not(@controlActExtension) or @controlActRoot" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:attribute name="validTimeLow" type="xsd:string" use="optional" />
    <xsd:attribute name="validTimeHigh" type="xsd:string" use="optional" />
    <xsd:attribute name="controlActRoot" type="Uid" use="optional" />
    <xsd:attribute name="controlActExtension" type="xsd:string" use="optional" />
  </xsd:complexType>
  <xsd:complexType name="BL.NONNULL">
```


© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="BL.NONNULL cannot be null">
          <sch:rule abstract="true" id="BL.NONNULL-0">
            <sch:assert test="not(@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="BL">
        <xsd:attribute name="nullFlavor" type="NullFlavor" use="prohibited" />
        <xsd:attribute name="value" type="xsd:boolean" use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="content is required if not null">
          <sch:rule abstract="true" id="ED-0">
            <sch:assert test="@nullFlavor or @value or xml or data" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="only one of value, data, xml">
          <sch:rule abstract="true" id="ED-1">
            <sch:assert test="count(*[self::value or self::xml or self::data])&#60;=1" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="integrityCheckAlgorithm required">
          <sch:rule abstract="true" id="ED-2">
            <sch:assert test="not(integrityCheck) or @integrityCheckAlgorithm" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="thumbnails do not use reference">
          <sch:rule abstract="true" id="ED-3">
            <sch:assert test="not(thumbnail) or thumbnail[not(@nullFlavor and reference)]" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="thumbnails do not have thumbnails">
          <sch:rule abstract="true" id="ED-4">
            <sch:assert test="not(thumbnail) or thumbnail[not(thumbnail)]" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="compression only on binary">
          <sch:rule abstract="true" id="ED-5">
            <sch:assert test="(@compression and data) or not(@compression or data)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="mediaType cannot be null">
          <sch:rule abstract="true" id="ED-6">
            <sch:assert test="@mediaType" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="value implies mediaType is text/plain">
          <sch:rule abstract="true" id="ED-7">
            <sch:assert test="not(@value) or (@value and (not(@mediaType) or
@mediaType=&#39;text/plain&#39;))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no charset for value or xml">
          <sch:rule abstract="true" id="ED-8">
            <sch:assert test="not(@value or xml) or not(@charset)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="ED-9">
            <sch:assert test="count(*[self::reference or self::thumbnail][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no nested translations">
```

© ISO 2007 – All rights reserved

```
          <sch:rule abstract="true" id="ED-10">
            <sch:assert test="not(translation) or thumbnail[not(translation)]" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value if null">
          <sch:rule abstract="true" id="ED-11">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@value))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no data if null">
          <sch:rule abstract="true" id="ED-12">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@data))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no xml if null">
          <sch:rule abstract="true" id="ED-13">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@xml))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no reference if null">
          <sch:rule abstract="true" id="ED-14">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@reference))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no mediaType if null">
          <sch:rule abstract="true" id="ED-15">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@mediaType))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no charset if null">
          <sch:rule abstract="true" id="ED-16">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@charset))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no language if null">
          <sch:rule abstract="true" id="ED-17">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@language))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no compression if null">
          <sch:rule abstract="true" id="ED-18">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@compression))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no integrityCheck if null">
          <sch:rule abstract="true" id="ED-19">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@integrityCheck))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no integrityCheckAlgorithm if null">
          <sch:rule abstract="true" id="ED-20">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and
not(@integrityCheckAlgorithm))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no thumbnail if null">
          <sch:rule abstract="true" id="ED-21">
            <sch:assert test="not(@nullFlavor) or (not(thumbnail) or thumbnail\@nullFlavor)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no translation if null">
          <sch:rule abstract="true" id="ED-22">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@translation))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="optional" />
        <xsd:attribute name="mediaType" type="xsd:string" default="text/plain" use="optional"
/>
        <xsd:attribute name="charset" type="Code" use="optional" />
        <xsd:attribute name="language" type="Code" use="optional" />
        <xsd:attribute name="compression" type="Compression" use="optional" />
        <xsd:attribute name="integrityCheckAlgorithm" type="IntegrityCheckAlgorithm"
use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TEL.URL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no use">
          <sch:rule abstract="true" id="TEL.URL-0">
            <sch:assert test="not(use)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="schemes">
          <sch:rule abstract="true" id="TEL.URL-1">
            <sch:assert test="(starts-with(@value, &#34;file&#34;) or starts-with(@value,
&#34;ftp&#34;) or starts-with(@value, &#34;http&#34;) or starts-with(@value, &#34;https&#34;)
or starts-with(@value, &#34;nfs&#34;))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="TEL">
        <xsd:sequence>
          <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="use" type="TelecommunicationAddressUse" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TEL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value is required">
          <sch:rule abstract="true" id="TEL-0">
            <sch:assert test="@nullFlavor or (not(@nullFlavor) and @value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value if null">
          <sch:rule abstract="true" id="TEL-1">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@value))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no use if null">
          <sch:rule abstract="true" id="TEL-2">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@use))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no useablePeriod if null">
          <sch:rule abstract="true" id="TEL-3">
            <sch:assert test="not(@nullFlavor) or (not(thumbnail) or
useablePeriod\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on TEL attributes">
          <sch:rule abstract="true" id="TEL-4">
```

© ISO 2007 – All rights reserved

```
            <sch:assert test="count(*[self::useablePeriod][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:anyURI" use="optional" />
        <xsd:attribute name="use" type="set_TelecommunicationAddressUse" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.TEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="text only">
          <sch:rule abstract="true" id="ED.TEXT-0">
            <sch:assert test="@mediaType = &#34;&#39;text/plain&#39;&#34;" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no xml">
          <sch:rule abstract="true" id="ED.TEXT-1">
            <sch:assert test="not(xml)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no data">
          <sch:rule abstract="true" id="ED.TEXT-2">
            <sch:assert test="not(data)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no integrityCheck">
          <sch:rule abstract="true" id="ED.TEXT-3">
            <sch:assert test="not(integrityCheck)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no thumbnail">
          <sch:rule abstract="true" id="ED.TEXT-4">
            <sch:assert test="not(thumbnail)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no compression">
          <sch:rule abstract="true" id="ED.TEXT-5">
            <sch:assert test="not(@compression)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="ED.TEXT-6">
            <sch:assert test="not(translation)" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="0" />
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="0" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="0" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="0" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="mediaType" type="xsd:string" fixed="text/plain" use="required" />
        <xsd:attribute name="compression" type="Compression" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ST">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no nested translations">
          <sch:rule abstract="true" id="ST-0">
            <sch:assert test="not(translation) or thumbnail[not(translation)]" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="null or content">
          <sch:rule abstract="true" id="ST-1">
            <sch:assert test="(@nullFlavor or (@value and string-length(@value)&#38;gt;0)) and
not(@nullFlavor and @value)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="translation" type="ST.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="optional" />
        <xsd:attribute name="language" type="Code" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ST.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="ST.NT-0">
            <sch:assert test="not(translation)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ST">
        <xsd:sequence>
          <xsd:element name="translation" type="ST.NT" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.IMAGE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="fixed to image">
          <sch:rule abstract="true" id="ED.IMAGE-0">
            <sch:assert test="starts-with(@mediaType, &#34;image/&#34;)" />
          </sch:rule>
        </sch:pattern>
```

© ISO 2007 – All rights reserved

```
        <sch:pattern name="no text">
          <sch:rule abstract="true" id="ED.IMAGE-1">
            <sch:assert test="not(@value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no xml">
          <sch:rule abstract="true" id="ED.IMAGE-2">
            <sch:assert test="not(xml)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="prohibited" />
        <xsd:attribute name="mediaType" type="xsd:string" use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.DOC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="mediaType">
          <sch:rule abstract="true" id="ED.DOC-0">
            <sch:assert test="mediaType = &#34;text/plain&#34; or mediaType =
&#34;text/html&#34; or mediaType = &#34;text/xml&#34; or mediaType =
&#34;application/pdf&#34;" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="mediaType" type="xsd:string" use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.DOC.REF">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="reference required">
          <sch:rule abstract="true" id="ED.DOC.REF-0">
            <sch:assert test="not(@nullFlavor) or (reference and not(reference/@nullFlavor))"
/>
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED.DOC">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="0" />
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="0" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="prohibited" />
        <xsd:attribute name="mediaType" type="xsd:string" use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.DOC.INLINE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no reference">
          <sch:rule abstract="true" id="ED.DOC.INLINE-0">
            <sch:assert test="not(reference)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED.DOC">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="1" />
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="mediaType" type="xsd:string" use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.SIGNATURE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no reference">
          <sch:rule abstract="true" id="ED.SIGNATURE-0">
            <sch:assert test="not(reference)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value">
          <sch:rule abstract="true" id="ED.SIGNATURE-1">
            <sch:assert test="not(@value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no data">
          <sch:rule abstract="true" id="ED.SIGNATURE-2">
            <sch:assert test="not(data)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no integrityCheck">
          <sch:rule abstract="true" id="ED.SIGNATURE-3">
            <sch:assert test="not(integrityCheck)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no thumbnail">
          <sch:rule abstract="true" id="ED.SIGNATURE-4">
            <sch:assert test="not(thumbnail)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no compression">
          <sch:rule abstract="true" id="ED.SIGNATURE-5">
            <sch:assert test="not(@compression)" />
          </sch:rule>
        </sch:pattern>
```

© ISO 2007 – All rights reserved

```
        <sch:pattern name="no language">
          <sch:rule abstract="true" id="ED.SIGNATURE-6">
            <sch:assert test="not(@language)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="mediaType">
          <sch:rule abstract="true" id="ED.SIGNATURE-7">
            <sch:assert test="@mediaType = &#34;&#39;text/xml&#39;&#34;" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="ED.SIGNATURE-8">
            <sch:assert test="not(translation)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="0" />
          <xsd:element name="xml" type="xsd:anyType" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="0" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="0" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="prohibited" />
        <xsd:attribute name="mediaType" type="xsd:string" fixed="text/xml" use="required" />
        <xsd:attribute name="language" type="Code" use="prohibited" />
        <xsd:attribute name="compression" type="Compression" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.STRUCTUREDTEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="mediatype">
          <sch:rule abstract="true" id="ED.STRUCTUREDTEXT-0">
            <sch:assert test="@mediaType = &#34;&#39;text/x-hl7-text+xml&#39;&#34;" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value">
          <sch:rule abstract="true" id="ED.STRUCTUREDTEXT-1">
            <sch:assert test="not(@value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no data">
          <sch:rule abstract="true" id="ED.STRUCTUREDTEXT-2">
            <sch:assert test="not(data)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="ED.STRUCTUREDTEXT-3">
            <sch:assert test="not(translation)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="0" />
          <xsd:element name="xml" type="StrucDoc.Text" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="0" />
```


© ISO 2007 – All rights reserved

```
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="prohibited" />
        <xsd:attribute name="mediaType" type="xsd:string" fixed="text/x-hl7-text+xml"
use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ED.STRUCTUREDTITLE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="mediatype">
          <sch:rule abstract="true" id="ED.STRUCTUREDTITLE-0">
            <sch:assert test="@mediaType = &#34;&#39;text/x-hl7-text+xml&#39;&#34;" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value">
          <sch:rule abstract="true" id="ED.STRUCTUREDTITLE-1">
            <sch:assert test="not(@value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no data">
          <sch:rule abstract="true" id="ED.STRUCTUREDTITLE-2">
            <sch:assert test="not(data)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="ED.STRUCTUREDTITLE-3">
            <sch:assert test="not(translation)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="ED">
        <xsd:sequence>
          <xsd:element name="data" type="xsd:base64Binary" minOccurs="0" maxOccurs="0" />
          <xsd:element name="xml" type="StrucDoc.Title" minOccurs="0" maxOccurs="1" />
          <xsd:element name="reference" type="TEL.URL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="integrityCheck" type="xsd:base64Binary" minOccurs="0"
maxOccurs="1" />
          <xsd:element name="thumbnail" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="description" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="ED" minOccurs="0" maxOccurs="0" />
          <xsd:element name="xml" type="StrucDoc.Title" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:string" use="prohibited" />
        <xsd:attribute name="mediaType" type="xsd:string" fixed="text/x-hl7-text+xml"
use="required" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.Title">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="footnote" type="StrucDoc.TitleFootnote" />
        <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
        <xsd:element name="br" type="StrucDoc.Br" />
        <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
        <xsd:element name="sub" type="StrucDoc.Sub" />
        <xsd:element name="sup" type="StrucDoc.Sup" />
        <xsd:element name="content" type="StrucDoc.CMTitle" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.TitleFootnote">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="footnote" type="StrucDoc.TitleFootnote" />
        <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" />
        <xsd:element name="br" type="StrucDoc.Br" />
        <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" />
```

© ISO 2007 – All rights reserved

```
        <xsd:element name="sub" type="StrucDoc.Sub" />
        <xsd:element name="sup" type="StrucDoc.Sup" />
        <xsd:element name="content" type="StrucDoc.CMTitle" />
      </xsd:choice>
    </xsd:sequence>
    <xsd:attributeGroup ref="StrucDoc.Base" />
  </xsd:complexType>
  <xsd:complexType mixed="true" name="StrucDoc.CMTitle">
    <xsd:sequence>
      <xsd:element name="footnote" type="StrucDoc.TitleFootnote" minOccurs="0" maxOccurs="1"
/>
      <xsd:element name="footnoteRef" type="StrucDoc.FootnoteRef" minOccurs="0" maxOccurs="1"
/>
      <xsd:element name="br" type="StrucDoc.Br" minOccurs="0" maxOccurs="1" />
      <xsd:element name="linkHtml" type="StrucDoc.LinkHtml" minOccurs="0" maxOccurs="1" />
      <xsd:element name="sub" type="StrucDoc.Sub" minOccurs="0" maxOccurs="1" />
      <xsd:element name="sup" type="StrucDoc.Sup" minOccurs="0" maxOccurs="1" />
      <xsd:element name="content" type="StrucDoc.CMTitle" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no code if no value">
          <sch:rule abstract="true" id="SC-0">
            <sch:assert test="not(@nullFlavor) or (not(code) or (code/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SC attributes">
          <sch:rule abstract="true" id="SC-1">
            <sch:assert test="count(*[self::code][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no originalText">
          <sch:rule abstract="true" id="SC-2">
            <sch:assert test="(not(originalText) or not(oroginalText/@nullFlavor)) or
(not(code) or (code/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ST">
        <xsd:sequence>
          <xsd:element name="code" type="CD" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="CD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or code and/or originalText">
          <sch:rule abstract="true" id="CD-0">
            <sch:assert test="@nullFlavor or @code or (originalText and
not(originalText/@nullFlavor) or (originalTextReference and
not(originalTextReference/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="other requires codeSystem or valueSet">
          <sch:rule abstract="true" id="CD-1">
            <sch:assert test="@nullFlavor != &#34;OTH&#34; or @codeSystem or @valueSet" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="code requires codeSystem">
          <sch:rule abstract="true" id="CD-2">
            <sch:assert test="@codeSystem or not(@code)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="codeSystemName only if codeSystem">
          <sch:rule abstract="true" id="CD-3">
```


© ISO 2007 – All rights reserved

```
            <sch:assert test="@codeSystem or not(@codeSystemName)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="codeSystemVersion only if codeSystem">
          <sch:rule abstract="true" id="CD-4">
            <sch:assert test="@codeSystem or not(@codeSystemVersion)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="displayName only if code">
          <sch:rule abstract="true" id="CD-5">
            <sch:assert test="@code or not(@displayName)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="valueSet requires valueSetVersion">
          <sch:rule abstract="true" id="CD-6">
            <sch:assert test="not(@valueSet) or (@valueSet and @valueSetVersion)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No original text on translations">
          <sch:rule abstract="true" id="CD-7">
            <sch:assert test="not translation/originalText" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="Translations cannot have translations">
          <sch:rule abstract="true" id="CD-8">
            <sch:assert test="not translation/translation" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on CD elements">
          <sch:rule abstract="true" id="CD-9">
            <sch:assert test="count(*[self::displayName or self::originalText or
self::originalTextReference or self::translation][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no code if null">
          <sch:rule abstract="true" id="CD-10">
            <sch:assert test="not(@nullFlavor) or (not(thumbnail) or code\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no displayName if null">
          <sch:rule abstract="true" id="CD-11">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@displayName))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no source if null">
          <sch:rule abstract="true" id="CD-12">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@source))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="displayName" type="ST" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="translation" type="CD" minOccurs="0" maxOccurs="unbounded" />
          <xsd:element name="source" type="XReference" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="code" type="xsd:string" use="optional" />
        <xsd:attribute name="codeSystem" type="Uid" use="optional" />
        <xsd:attribute name="codeSystemName" type="xsd:string" use="optional" />
        <xsd:attribute name="codeSystemVersion" type="xsd:string" use="optional" />
        <xsd:attribute name="valueSet" type="xsd:string" use="optional" />
        <xsd:attribute name="valueSetVersion" type="xsd:string" use="optional" />
        <xsd:attribute name="codingRationale" type="set_CodingRationale" use="optional" />
        <xsd:attribute name="id" type="xsd:ID" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="XReference">
```

© ISO 2007 – All rights reserved

```
    <xsd:attribute name="xref" type="xsd:IDREF" use="required" />
  </xsd:complexType>
  <xsd:complexType name="SC.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no translations">
          <sch:rule abstract="true" id="SC.NT-0">
            <sch:assert test="not(translation)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="SC">
        <xsd:sequence>
          <xsd:element name="translation" type="ST.NT" minOccurs="0" maxOccurs="0" />
          <xsd:element name="code" type="CD" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TEL.EMAIL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="email only">
          <sch:rule abstract="true" id="TEL.EMAIL-0">
            <sch:assert test="starts-with(@value, &#34;mailto&#34;)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="TEL.PERSON">
        <xsd:sequence>
          <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TEL.PERSON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="Personal Address">
          <sch:rule abstract="true" id="TEL.PERSON-0">
            <sch:assert test="starts-with(@value, &#34;tel&#34;) or starts-with(@value,
&#34;x-text-fax&#34;) or starts-with(@value, &#34;x-text-tel&#34;) or starts-with(@value,
&#34;mailto&#34;)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="TEL">
        <xsd:sequence>
          <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TEL.PHONE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="Phone">
          <sch:rule abstract="true" id="TEL.PHONE-0">
            <sch:assert test="starts-with(@value, &#34;tel&#34;) or starts-with(@value,
&#34;x-text-fax&#34;) or starts-with(@value, &#34;x-text-tel&#34;)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="TEL.PERSON">
```


© ISO 2007 – All rights reserved

```
      <xsd:sequence>
        <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="II">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="root is required">
        <sch:rule abstract="true" id="II-0">
          <sch:assert test="@nullFlavor or (not(@nullFlavor) and @root)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no root if null">
        <sch:rule abstract="true" id="II-1">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@root))" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no extension if null">
        <sch:rule abstract="true" id="II-2">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@extension))" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no identifierName if null">
        <sch:rule abstract="true" id="II-3">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@identifierName))" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no displayable if null">
        <sch:rule abstract="true" id="II-4">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@displayable))" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no reliability if null">
        <sch:rule abstract="true" id="II-5">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@reliability))" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:attribute name="root" type="Uid" use="optional" />
      <xsd:attribute name="extension" type="xsd:string" use="optional" />
      <xsd:attribute name="identifierName" type="xsd:string" use="optional" />
      <xsd:attribute name="displayable" type="xsd:boolean" use="optional" />
      <xsd:attribute name="scope" type="IdentifierScope" use="optional" />
      <xsd:attribute name="reliability" type="IdentifierReliability" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="CD.CV">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="no translations">
        <sch:rule abstract="true" id="CD.CV-0">
          <sch:assert test="not(translation)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no source">
        <sch:rule abstract="true" id="CD.CV-1">
          <sch:assert test="not(@source)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="CD">
      <xsd:sequence>
        <xsd:element name="displayName" type="ST" minOccurs="0" maxOccurs="1" />
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="translation" type="CD" minOccurs="0" maxOccurs="0" />
          <xsd:element name="source" type="XReference" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have a code or a value">
          <sch:rule abstract="true" id="CO-0">
            <sch:assert test="@nullFlavor or (@value or (code and not(code/@nullFlavor)))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="CO-1">
            <sch:assert test="count(*[self::code][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no value if null">
          <sch:rule abstract="true" id="CO-2">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@value))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no code if null">
          <sch:rule abstract="true" id="CO-3">
            <sch:assert test="not(@nullFlavor) or (not code or code/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="code" type="CD" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QTY">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="uncertainty">
          <sch:rule abstract="true" id="QTY-0">
            <sch:assert test="not(uncertainty/@expression) and not(uncertainty/@uncertainty)
and not(uncertainty/@originalText)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QTY-1">
            <sch:assert test="count(*[self::originalText or self::uncertainty or
self::expression][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension
or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="uncertaintyType" type="UncertaintyType" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="CS">
    <xsd:annotation>
```


© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="code is required">
          <sch:rule abstract="true" id="CS-0">
            <sch:assert test="@nullFlavor or (not(@nullFlavor) and @code)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no code if null">
          <sch:rule abstract="true" id="CS-1">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@code))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:attribute name="code" type="xsd:string" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="AD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or parts">
          <sch:rule abstract="true" id="AD-0">
            <sch:assert test="(@nullFlavor or part) and not(@nullFlavor and part)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no useablePeriod if null">
          <sch:rule abstract="true" id="AD-1">
            <sch:assert test="not(@nullFlavor) or (not(thumbnail) or
useablePeriod\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on AD attributes">
          <sch:rule abstract="true" id="AD-2">
            <sch:assert test="count(*[self::useablePeriod][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="part" type="ADXP" />
            <xsd:element name="br" type="ADXP.BR" />
            <xsd:element name="addressLine" type="ADXP.AL" />
            <xsd:element name="additionalLocator" type="ADXP.ADL" />
            <xsd:element name="unitID" type="ADXP.UNID" />
            <xsd:element name="unitType" type="ADXP.UNIT" />
            <xsd:element name="deliveryAddressLine" type="ADXP.DAL" />
            <xsd:element name="deliveryInstallationType" type="ADXP.DINST" />
            <xsd:element name="deliveryInstallationArea" type="ADXP.DINSTA" />
            <xsd:element name="deliveryInstallationQualifier" type="ADXP.DINSTQ" />
            <xsd:element name="deliveryMode" type="ADXP.DMOD" />
            <xsd:element name="deliveryModeIdentifier" type="ADXP.DMODID" />
            <xsd:element name="streetAddressLine" type="ADXP.SAL" />
            <xsd:element name="houseNumber" type="ADXP.BNR" />
            <xsd:element name="houseNumberNumeric" type="ADXP.BNN" />
            <xsd:element name="buildingNumberSuffix" type="ADXP.BNS" />
            <xsd:element name="streetName" type="ADXP.STR" />
            <xsd:element name="streetNameBase" type="ADXP.STB" />
            <xsd:element name="streetType" type="ADXP.STTYP" />
            <xsd:element name="intersection" type="ADXP.INT" />
            <xsd:element name="direction" type="ADXP.DIR" />
            <xsd:element name="careOf" type="ADXP.CAR" />
            <xsd:element name="censusTract" type="ADXP.CEN" />
            <xsd:element name="country" type="ADXP.CNT" />
            <xsd:element name="county" type="ADXP.CPA" />
            <xsd:element name="city" type="ADXP.CTY" />
            <xsd:element name="postBox" type="ADXP.POB" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="precinct" type="ADXP.PRE" />
          <xsd:element name="state" type="ADXP.STA" />
          <xsd:element name="postalCode" type="ADXP.ZIP" />
        </xsd:choice>
        <xsd:element name="useablePeriod" type="QSET_TS" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="use" type="set_PostalAddressUse" use="optional" />
      <xsd:attribute name="isNotOrdered" type="xsd:boolean" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType mixed="true" name="ADXP">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="value is required">
        <sch:rule abstract="true" id="ADXP-0">
          <sch:assert test="string-length(@value) &#62; 0" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="code requires codeSystem">
        <sch:rule abstract="true" id="ADXP-1">
          <sch:assert test="@codeSystem or not(@code)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="codeSystemVersion only if codeSystem">
        <sch:rule abstract="true" id="ADXP-2">
          <sch:assert test="@codeSystem or not(@codeSystemVersion)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:attribute name="code" type="xsd:string" use="optional" />
  <xsd:attribute name="codeSystem" type="xsd:string" use="optional" />
  <xsd:attribute name="codeSystemVersion" type="xsd:string" use="optional" />
  <xsd:attribute name="language" type="Code" use="optional" />
  <xsd:attribute name="type" type="AddressPartType" use="optional" />
</xsd:complexType>
<xsd:complexType name="ADXP.BR">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DEL" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.AL">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="AL" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.ADL">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="ADL" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.UNID">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="UNID" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.UNIT">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="UNIT" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```


© ISO 2007 – All rights reserved

```
<xsd:complexType name="ADXP.DAL">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DAL" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DINST">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DINST" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DINSTA">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DINSTA" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DINSTQ">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DINSTQ" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DMOD">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DMOD" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DMODID">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DMODID" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.SAL">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="SAL" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.BNR">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="BNR" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.BNN">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="BNN" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.BNS">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="BNS" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.STR">
  <xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="STR" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.STB">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="STB" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.STTYP">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="STTYP" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.INT">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="INT" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.DIR">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="DIR" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.CAR">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="CAR" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.CEN">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="CEN" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.CNT">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="CNT" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.CPA">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="CPA" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.CTY">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="CTY" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ADXP.POB">
  <xsd:complexContent>
    <xsd:restriction base="ADXP">
      <xsd:attribute name="type" type="AddressPartType" fixed="POB" />
```


© ISO 2007 – All rights reserved

```
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ADXP.PRE">
    <xsd:complexContent>
      <xsd:restriction base="ADXP">
        <xsd:attribute name="type" type="AddressPartType" fixed="PRE" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ADXP.STA">
    <xsd:complexContent>
      <xsd:restriction base="ADXP">
        <xsd:attribute name="type" type="AddressPartType" fixed="STA" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ADXP.ZIP">
    <xsd:complexContent>
      <xsd:restriction base="ADXP">
        <xsd:attribute name="type" type="AddressPartType" fixed="ZIP" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or parts">
          <sch:rule abstract="true" id="EN-0">
            <sch:assert test="(@nullFlavor or part) and not(@nullFlavor and part)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="part" type="ENXP" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="use" type="set_EntityNameUse" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ENXP">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value is required">
          <sch:rule abstract="true" id="ENXP-0">
            <sch:assert test="string-length(@value) &#62; 0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="code requires codeSystem">
          <sch:rule abstract="true" id="ENXP-1">
            <sch:assert test="@codeSystem or not(@code)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="codeSystemVersion only if codeSystem">
          <sch:rule abstract="true" id="ENXP-2">
            <sch:assert test="@codeSystem or not(@codeSystemVersion)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:attribute name="value" type="xsd:string" use="optional" />
    <xsd:attribute name="code" type="xsd:string" use="optional" />
    <xsd:attribute name="codeSystem" type="xsd:string" use="optional" />
    <xsd:attribute name="codeSystemVersion" type="xsd:string" use="optional" />
    <xsd:attribute name="language" type="Code" use="optional" />
    <xsd:attribute name="type" type="EntityNamePartType" use="optional" />
    <xsd:attribute name="qualifier" type="set_EntityNamePartQualifier" use="optional" />
  </xsd:complexType>
  <xsd:complexType name="EN.TN">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="only one part with no type">
          <sch:rule abstract="true" id="EN.TN-0">
            <sch:assert test="@nullFlavor or (count(part) = 1 and not part/@type)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="EN" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EN.PN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no parts are qualified by LS">
          <sch:rule abstract="true" id="EN.PN-0">
            <sch:assert test="count(part[contains(@qualifier, &#39;LS&#39;)])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="EN" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EN.ON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no parts are person types">
          <sch:rule abstract="true" id="EN.ON-0">
            <sch:assert test="count(part[type = &#39;FAM&#39; or type = &#39;GIV&#39;])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="EN" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or value">
          <sch:rule abstract="true" id="TS-0">
            <sch:assert test="(@nullFlavor or @value) and not(@nullFlavor and @value)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))((0[1-9])|([1-2][0-
9]|3[0-1]))([0-1][0-9]|2[0-3])([0-5][0-9]([0-5][0-9](\.[0-9]{1,4})?)?)?)?)?)?([+\-](0[0-
9]|1[0-3])([0-5][0-9]))?" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS.DATE">
    <xsd:complexContent>
      <xsd:restriction base="TS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
        </xsd:sequence>
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}((0[1-9])|(1[0-2])((0[1-9])|([1-2][0-9]|3[0-
1]))?)?" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS.DATE.FULL">
    <xsd:complexContent>
      <xsd:restriction base="TS.DATE">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}(0[1-9])|(1[0-2]))(0[1-9])|([1-2][0-9]|3[0-
1])" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS.DATETIME">
    <xsd:complexContent>
      <xsd:restriction base="TS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))((0[1-9])|([1-2][0-
9]|3[0-1]))(([0-1][0-9]|2[0-3])([0-5][0-9]([0-5][0-9])?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-
9]))?" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS.DATETIME.FULL">
    <xsd:complexContent>
      <xsd:restriction base="TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}(0[1-9])|(1[0-2])(0[1-9])|([1-2][0-9]|3[0-
1])([0-1][0-9]|2[0-3])[0-5][0-9][0-5][0-9]([+\-](0[0-9]|1[0-3])([0-5][0-9]))?" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TS.BIRTH">
    <xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
      <xsd:restriction base="TS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:pattern value="[1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(0[1-9])|([1-2][0-9]|3[0-
1])(([0-1][0-9]|2[0-3])([0-5][0-9]([0-5][0-9]))[+\-](0[0-9]|1[0-3])([0-5][0-9]))?)?" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="numerator and denominator required">
          <sch:rule abstract="true" id="RTO-0">
            <sch:assert test="@nullFlavor or ((numerator and not(numerator/@nullFlavor)) and
(numerator and not(numerator/@nullFlavor)))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on RTO Attributes">
          <sch:rule abstract="true" id="RTO-1">
            <sch:assert test="count(*[self::numerator or self::denominator][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no uncertainty">
          <sch:rule abstract="true" id="RTO-2">
            <sch:assert test="not(uncertainty)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="numerator" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="denominator" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or value">
          <sch:rule abstract="true" id="MO-0">
            <sch:assert test="(@nullFlavor or @value) and not(@nullFlavor and @value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="null or currency">
          <sch:rule abstract="true" id="MO-1">
            <sch:assert test="(@nullFlavor or @currency) and not(@nullFlavor and @currency)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no precision if null">
          <sch:rule abstract="true" id="MO-2">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@precision))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:attribute name="value" type="xsd:double" use="optional" />
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="precision" type="xsd:int" use="optional" />
        <xsd:attribute name="currency" type="Code" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no translation if null">
          <sch:rule abstract="true" id="PQ-0">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@translation))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="PQV">
        <xsd:sequence>
          <xsd:element name="translation" type="PQR" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="unit" type="Code" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PQV">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or value">
          <sch:rule abstract="true" id="PQV-0">
            <sch:assert test="(@nullFlavor or @value) and not(@nullFlavor and @value)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no precision if null">
          <sch:rule abstract="true" id="PQV-1">
            <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@precision))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="source" type="XReference" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="value" type="xsd:double" use="optional" />
        <xsd:attribute name="precision" type="xsd:int" default="0" use="optional" />
        <xsd:attribute name="codingRationale" type="set_CodingRationale" use="optional" />
        <xsd:attribute name="id" type="xsd:ID" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PQR">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or unit">
          <sch:rule abstract="true" id="PQR-0">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no originalText">
          <sch:rule abstract="true" id="PQR-1">
            <sch:assert test="not(@originalText)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PQR">
          <sch:rule abstract="true" id="PQR-2">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and
not(@controlActRoot) and not(@controlActExtension) and not(@updateMode)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```

© ISO 2007 – All rights reserved

```
  <xsd:complexContent>
    <xsd:extension base="PQV">
      <xsd:sequence>
        <xsd:element name="unit" type="CD.CV" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PQ.TIME">
  <xsd:complexContent>
    <xsd:restriction base="PQ">
      <xsd:sequence>
        <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="source" type="XReference" minOccurs="0" maxOccurs="1" />
        <xsd:element name="translation" type="PQR" minOccurs="0" maxOccurs="0" />
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="REAL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or value">
        <sch:rule abstract="true" id="REAL-0">
          <sch:assert test="(@nullFlavor or @value) and not(@nullFlavor and @value)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="no precision if null">
        <sch:rule abstract="true" id="REAL-1">
          <sch:assert test="not(@nullFlavor) or (@nullFlavor and not(@precision))" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QTY">
      <xsd:attribute name="value" type="xsd:double" use="optional" />
      <xsd:attribute name="precision" type="xsd:int" default="0" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="INT">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="no uncertainty">
        <sch:rule abstract="true" id="INT-0">
          <sch:assert test="not(uncertainty)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="null or value">
        <sch:rule abstract="true" id="INT-1">
          <sch:assert test="(@nullFlavor or @value) and not(@nullFlavor and @value)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QTY">
      <xsd:attribute name="value" type="xsd:int" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="INT.NONNEG">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not negative">
        <sch:rule abstract="true" id="INT.NONNEG-0">
          <sch:assert test="(@nullFlavor) or (@value &#62;= 0)" />
        </sch:rule>
      </sch:pattern>
```


© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="INT">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="INT.POS">
    <xsd:complexContent>
      <xsd:restriction base="INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_BL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_BL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_BL">
        <xsd:sequence>
          <xsd:element name="item" type="BL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_BL">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_BL.NONNULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_BL.NONNULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_BL.NONNULL">
        <xsd:sequence>
          <xsd:element name="item" type="BL.NONNULL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_BL.NONNULL">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_ED">
    <xsd:annotation>
      <xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED">
      <xsd:sequence>
        <xsd:element name="item" type="ED" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.IMAGE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.IMAGE-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED.IMAGE">
      <xsd:sequence>
        <xsd:element name="item" type="ED.IMAGE" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED.IMAGE">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.TEXT">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.TEXT-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED.TEXT">
      <xsd:sequence>
        <xsd:element name="item" type="ED.TEXT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED.TEXT">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.DOC">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.DOC-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
```


© ISO 2007 – All rights reserved

```
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED.DOC">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED.DOC">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.DOC.REF">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.DOC.REF-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED.DOC.REF">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC.REF" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED.DOC.REF">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.DOC.INLINE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.DOC.INLINE-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_ED.DOC.INLINE">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC.INLINE" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_ED.DOC.INLINE">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_ED.SIGNATURE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_ED.SIGNATURE-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.SIGNATURE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.SIGNATURE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_ED.SIGNATURE">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_ED.STRUCTUREDTEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_ED.STRUCTUREDTEXT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.STRUCTUREDTEXT">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTEXT" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_ED.STRUCTUREDTEXT">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_ED.STRUCTUREDTITLE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_ED.STRUCTUREDTITLE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.STRUCTUREDTITLE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTITLE" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_ED.STRUCTUREDTITLE">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_ST">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_ST-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="COLL_ST">
        <xsd:sequence>
          <xsd:element name="item" type="ST" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_ST">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_ST.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_ST.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_ST.NT">
        <xsd:sequence>
          <xsd:element name="item" type="ST.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_ST.NT">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_SC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_SC-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_SC">
        <xsd:sequence>
          <xsd:element name="item" type="SC" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_SC">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_SC.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_SC.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_SC.NT">
        <xsd:sequence>
```

© ISO 2007 – All rights reserved

```
        <xsd:element name="item" type="SC.NT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_SC.NT">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TEL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TEL-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL">
      <xsd:sequence>
        <xsd:element name="item" type="TEL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_TEL">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TEL.EMAIL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TEL.EMAIL-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.EMAIL">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.EMAIL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_TEL.EMAIL">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TEL.PERSON">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TEL.PERSON-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.PERSON">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.PERSON" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
```


© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TEL.PERSON">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_TEL.PHONE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_TEL.PHONE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_TEL.PHONE">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.PHONE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TEL.PHONE">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_TEL.URL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_TEL.URL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_TEL.URL">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.URL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TEL.URL">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_II">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_II-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_II">
        <xsd:sequence>
          <xsd:element name="item" type="II" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_II">
```

© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_CD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_CD-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_CD">
        <xsd:sequence>
          <xsd:element name="item" type="CD" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_CD">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_CD.CV">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_CD.CV-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_CD.CV">
        <xsd:sequence>
          <xsd:element name="item" type="CD.CV" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_CD.CV">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_CO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_CO">
        <xsd:sequence>
          <xsd:element name="item" type="CO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_CO">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
</xsd:complexType>
<xsd:complexType name="DSET_CS">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_CS-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_CS">
      <xsd:sequence>
        <xsd:element name="item" type="CS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_CS">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_AD">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_AD-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_AD">
      <xsd:sequence>
        <xsd:element name="item" type="AD" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_AD">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_EN">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_EN-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_EN">
      <xsd:sequence>
        <xsd:element name="item" type="EN" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_EN">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_EN.TN">
  <xsd:annotation>
```

© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_EN.TN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_EN.TN">
        <xsd:sequence>
          <xsd:element name="item" type="EN.TN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_EN.TN">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_EN.PN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_EN.PN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_EN.PN">
        <xsd:sequence>
          <xsd:element name="item" type="EN.PN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_EN.PN">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_EN.ON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_EN.ON-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_EN.ON">
        <xsd:sequence>
          <xsd:element name="item" type="EN.ON" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_EN.ON">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_TS-0">
```


© ISO 2007 – All rights reserved

```
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TS">
      <xsd:sequence>
        <xsd:element name="item" type="TS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_TS">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TS.DATE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TS.DATE-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TS.DATE">
      <xsd:sequence>
        <xsd:element name="item" type="TS.DATE" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_TS.DATE">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TS.DATE.FULL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TS.DATE.FULL-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_TS.DATE.FULL">
      <xsd:sequence>
        <xsd:element name="item" type="TS.DATE.FULL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_TS.DATE.FULL">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_TS.DATETIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_TS.DATETIME-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
```

© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TS.DATETIME">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_TS.DATETIME.FULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TS.DATETIME.FULL">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_TS.BIRTH-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="item" type="TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_TS.BIRTH">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_RTO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="COLL_RTO">
        <xsd:sequence>
          <xsd:element name="item" type="RTO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_RTO">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_MO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_MO">
        <xsd:sequence>
          <xsd:element name="item" type="MO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_MO">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_PQ-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_PQ">
        <xsd:sequence>
          <xsd:element name="item" type="PQ" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_PQ">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_PQ.TIME-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_PQ.TIME">
        <xsd:sequence>
```

© ISO 2007 – All rights reserved

```
        <xsd:element name="item" type="PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_PQ.TIME">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_REAL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_REAL-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_REAL">
      <xsd:sequence>
        <xsd:element name="item" type="REAL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_REAL">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_INT">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_INT-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_INT">
      <xsd:sequence>
        <xsd:element name="item" type="INT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType abstract="true" name="COLL_INT">
  <xsd:complexContent>
    <xsd:extension base="ANY" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="DSET_INT.NONNEG">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="DSET_INT.NONNEG-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="COLL_INT.NONNEG">
      <xsd:sequence>
        <xsd:element name="item" type="INT.NONNEG" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
```


© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_INT.NONNEG">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DSET_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="DSET_INT.POS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="COLL_INT.POS">
        <xsd:sequence>
          <xsd:element name="item" type="INT.POS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="COLL_INT.POS">
    <xsd:complexContent>
      <xsd:extension base="ANY" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_BL">
    <xsd:complexContent>
      <xsd:extension base="COLL_BL">
        <xsd:sequence>
          <xsd:element name="item" type="BL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_BL.NONNULL">
    <xsd:complexContent>
      <xsd:extension base="COLL_BL.NONNULL">
        <xsd:sequence>
          <xsd:element name="item" type="BL.NONNULL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED">
        <xsd:sequence>
          <xsd:element name="item" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.IMAGE">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.IMAGE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.IMAGE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.TEXT">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.TEXT">
        <xsd:sequence>
          <xsd:element name="item" type="ED.TEXT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.DOC">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.DOC">
        <xsd:sequence>
          <xsd:element name="item" type="ED.DOC" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.DOC.REF">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.DOC.REF">
        <xsd:sequence>
          <xsd:element name="item" type="ED.DOC.REF" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.DOC.INLINE">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.DOC.INLINE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.DOC.INLINE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.SIGNATURE">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.SIGNATURE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.SIGNATURE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.STRUCTUREDTEXT">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.STRUCTUREDTEXT">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTEXT" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ED.STRUCTUREDTITLE">
    <xsd:complexContent>
      <xsd:extension base="COLL_ED.STRUCTUREDTITLE">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTITLE" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ST">
    <xsd:complexContent>
      <xsd:extension base="COLL_ST">
        <xsd:sequence>
          <xsd:element name="item" type="ST" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_ST.NT">
    <xsd:complexContent>
      <xsd:extension base="COLL_ST.NT">
        <xsd:sequence>
```


© ISO 2007 – All rights reserved

```
        <xsd:element name="item" type="ST.NT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_SC">
  <xsd:complexContent>
    <xsd:extension base="COLL_SC">
      <xsd:sequence>
        <xsd:element name="item" type="SC" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_SC.NT">
  <xsd:complexContent>
    <xsd:extension base="COLL_SC.NT">
      <xsd:sequence>
        <xsd:element name="item" type="SC.NT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_TEL">
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL">
      <xsd:sequence>
        <xsd:element name="item" type="TEL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_TEL.EMAIL">
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.EMAIL">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.EMAIL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_TEL.PERSON">
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.PERSON">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.PERSON" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_TEL.PHONE">
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.PHONE">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.PHONE" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_TEL.URL">
  <xsd:complexContent>
    <xsd:extension base="COLL_TEL.URL">
      <xsd:sequence>
        <xsd:element name="item" type="TEL.URL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_II">
  <xsd:complexContent>
    <xsd:extension base="COLL_II">
      <xsd:sequence>
```

© ISO 2007 – All rights reserved

```
        <xsd:element name="item" type="II" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_CD">
  <xsd:complexContent>
    <xsd:extension base="COLL_CD">
      <xsd:sequence>
        <xsd:element name="item" type="CD" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_CD.CV">
  <xsd:complexContent>
    <xsd:extension base="COLL_CD.CV">
      <xsd:sequence>
        <xsd:element name="item" type="CD.CV" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_CO">
  <xsd:complexContent>
    <xsd:extension base="COLL_CO">
      <xsd:sequence>
        <xsd:element name="item" type="CO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_CS">
  <xsd:complexContent>
    <xsd:extension base="COLL_CS">
      <xsd:sequence>
        <xsd:element name="item" type="CS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_AD">
  <xsd:complexContent>
    <xsd:extension base="COLL_AD">
      <xsd:sequence>
        <xsd:element name="item" type="AD" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_EN">
  <xsd:complexContent>
    <xsd:extension base="COLL_EN">
      <xsd:sequence>
        <xsd:element name="item" type="EN" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_EN.TN">
  <xsd:complexContent>
    <xsd:extension base="COLL_EN.TN">
      <xsd:sequence>
        <xsd:element name="item" type="EN.TN" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_EN.PN">
  <xsd:complexContent>
    <xsd:extension base="COLL_EN.PN">
      <xsd:sequence>
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="item" type="EN.PN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_EN.ON">
    <xsd:complexContent>
      <xsd:extension base="COLL_EN.ON">
        <xsd:sequence>
          <xsd:element name="item" type="EN.ON" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS">
        <xsd:sequence>
          <xsd:element name="item" type="TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS.DATE">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATE">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS.DATE.FULL">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATE.FULL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS.DATETIME">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS.DATETIME.FULL">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_TS.BIRTH">
    <xsd:complexContent>
      <xsd:extension base="COLL_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="item" type="TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="BAG_RTO">
    <xsd:complexContent>
      <xsd:extension base="COLL_RTO">
```

© ISO 2007 – All rights reserved

```
      <xsd:sequence>
        <xsd:element name="item" type="RTO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_MO">
  <xsd:complexContent>
    <xsd:extension base="COLL_MO">
      <xsd:sequence>
        <xsd:element name="item" type="MO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_PQ">
  <xsd:complexContent>
    <xsd:extension base="COLL_PQ">
      <xsd:sequence>
        <xsd:element name="item" type="PQ" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_PQ.TIME">
  <xsd:complexContent>
    <xsd:extension base="COLL_PQ.TIME">
      <xsd:sequence>
        <xsd:element name="item" type="PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_REAL">
  <xsd:complexContent>
    <xsd:extension base="COLL_REAL">
      <xsd:sequence>
        <xsd:element name="item" type="REAL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_INT">
  <xsd:complexContent>
    <xsd:extension base="COLL_INT">
      <xsd:sequence>
        <xsd:element name="item" type="INT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_INT.NONNEG">
  <xsd:complexContent>
    <xsd:extension base="COLL_INT.NONNEG">
      <xsd:sequence>
        <xsd:element name="item" type="INT.NONNEG" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="BAG_INT.POS">
  <xsd:complexContent>
    <xsd:extension base="COLL_INT.POS">
      <xsd:sequence>
        <xsd:element name="item" type="INT.POS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_BL">
  <xsd:annotation>
    <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_BL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="BL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_BL.NONNULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_BL.NONNULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="BL.NONNULL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ED">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ED-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ED.IMAGE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ED.IMAGE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ED.IMAGE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ED.TEXT">
    <xsd:annotation>
      <xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_ED.TEXT-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="ED.TEXT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_ED.DOC">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_ED.DOC-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_ED.DOC.REF">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_ED.DOC.REF-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC.REF" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_ED.DOC.INLINE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_ED.DOC.INLINE-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="ED.DOC.INLINE" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_ED.SIGNATURE">
  <xsd:annotation>
    <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ED.SIGNATURE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ED.SIGNATURE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ED.STRUCTUREDTEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ED.STRUCTUREDTEXT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTEXT" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ED.STRUCTUREDTITLE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ED.STRUCTUREDTITLE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ED.STRUCTUREDTITLE" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ST">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ST-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ST" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_ST.NT">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_ST.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="ST.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_SC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_SC-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="SC" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_SC.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_SC.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="SC.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TEL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TEL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TEL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TEL.EMAIL">
```


© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TEL.EMAIL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.EMAIL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TEL.PERSON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TEL.PERSON-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.PERSON" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TEL.PHONE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TEL.PHONE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.PHONE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TEL.URL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TEL.URL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TEL.URL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_II">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_II-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="II" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_CD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_CD-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="CD" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_CD.CV">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_CD.CV-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="CD.CV" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_CO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="CO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_CS">
```


© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_CS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="CS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_AD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_AD-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="AD" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_EN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_EN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="EN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_EN.TN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_EN.TN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="EN.TN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_EN.PN">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_EN.PN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="EN.PN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_EN.ON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_EN.ON-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="EN.ON" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS.DATE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS.DATE.FULL">
```


© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS.DATE.FULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATE.FULL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS.DATETIME-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS.DATETIME.FULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="LIST_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null or not empty">
          <sch:rule abstract="true" id="LIST_TS.BIRTH-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

© ISO 2007 – All rights reserved

```
<xsd:complexType name="LIST_RTO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_RTO-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="RTO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_MO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_MO-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="MO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_PQ">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_PQ-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="PQ" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_PQ.TIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_PQ.TIME-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```


© ISO 2007 – All rights reserved

```
<xsd:complexType name="LIST_REAL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_REAL-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="REAL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_INT">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_INT-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="INT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_INT.NONNEG">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_INT.NONNEG-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="INT.NONNEG" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="LIST_INT.POS">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="null or not empty">
        <sch:rule abstract="true" id="LIST_INT.POS-0">
          <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="ANY">
      <xsd:sequence>
        <xsd:element name="item" type="INT.POS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

© ISO 2007 – All rights reserved

```
  <xsd:complexType name="IVL_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_CO-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_CO-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_CO-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_CO-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_CO">
        <xsd:sequence>
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_CO-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS.DATE-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS.DATE-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS.DATE-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS.DATE-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```

© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS.DATE-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS.DATE.FULL-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS.DATE.FULL-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS.DATE.FULL-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS.DATE.FULL-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS.DATE.FULL-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS.DATETIME-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS.DATETIME-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS.DATETIME-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS.DATETIME-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS.DATETIME">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS.DATETIME-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS.DATETIME.FULL-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS.DATETIME.FULL-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS.DATETIME.FULL-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS.DATETIME.FULL-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS.DATETIME.FULL-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_TS.BIRTH-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_TS.BIRTH-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_TS.BIRTH-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_TS.BIRTH-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_TS.BIRTH-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_RTO-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_RTO-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_RTO-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_RTO-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_RTO-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_MO">
    <xsd:annotation>
      <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_MO-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_MO-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_MO-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_MO-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_MO">
        <xsd:sequence>
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_MO-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_PQ-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
```

© ISO 2007 – All rights reserved

```
          <sch:rule abstract="true" id="IVL_PQ-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_PQ-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_PQ-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ">
        <xsd:sequence>
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_PQ-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_PQ.TIME-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_PQ.TIME-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
```


© ISO 2007 – All rights reserved

```
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_PQ.TIME-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_PQ.TIME-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_PQ.TIME-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_REAL-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_REAL-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_REAL-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
```

© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_REAL-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_REAL-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_INT-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_INT-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_INT-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_INT-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT">
        <xsd:sequence>
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_INT-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_INT.NONNEG-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_INT.NONNEG-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_INT.NONNEG-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_INT.NONNEG-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_INT.NONNEG-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="null rules">
          <sch:rule abstract="true" id="IVL_INT.POS-0">
            <sch:assert test="(@nullFlavor and not(any|low|high|width)) or (not(@nullFlavor)
and (any|low|high|width))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="co-occurence rules">
          <sch:rule abstract="true" id="IVL_INT.POS-1">
            <sch:assert test="(((any and not(any/@nullFlavor)) and not((low and
not(low/@nullFlavor)) or (high and not(high/@nullFlavor)) or (width and
not(width/@nullFlavor)))) or (not(any) or any/@nullFlavor)) and ((any and
not(any/@nullFlavor)) or (not(any) or any/@nullFlavor) and (((width and
not(width/@nullFlavor)) and (not(low) or low/@nullFlavor) and (not(high) or
high/@nullFlavor))) or ((not(width) or width/@nullFlavor) and ((low and not(low/@nullFlavor))
or (high and not(high/@nullFlavor))))))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="closed attributes only if limited">
          <sch:rule abstract="true" id="IVL_INT.POS-2">
            <sch:assert test="not(low and @lowClosed) and not(high and @highClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on IVL attributes">
          <sch:rule abstract="true" id="IVL_INT.POS-3">
            <sch:assert test="count(*[self::low or self::high or self::width][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.POS">
        <xsd:sequence>
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
      </xsd:extension>
```


© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType abstract="true" name="QSET_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="QSET_INT.POS-0">
            <sch:assert test="count(*[self::originalText][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_CO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_CO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_CO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="0" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATE-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATE-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATE">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATE.FULL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATE.FULL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATETIME-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATETIME-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATETIME.FULL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS.DATETIME.FULL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_TS.BIRTH-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_TS.BIRTH-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_RTO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_RTO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_RTO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_MO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_MO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_MO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_PQ-0">
```


© ISO 2007 – All rights reserved

```
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_PQ-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_PQ">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_PQ.TIME-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_PQ.TIME-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_REAL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_REAL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_REAL">
        <xsd:sequence>
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_INT-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_INT-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.LOW_INT.NONNEG-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_INT.NONNEG-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.LOW_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
```


© ISO 2007 – All rights reserved

```
          <sch:rule abstract="true" id="IVL.LOW_INT.POS-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.LOW_INT.POS-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT.POS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.WIDTH_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="width">
          <sch:rule abstract="true" id="IVL.WIDTH_CO-0">
            <sch:assert test="width and not(width\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.WIDTH_CO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.WIDTH_CO-2">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_CO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.WIDTH_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="width">
          <sch:rule abstract="true" id="IVL.WIDTH_TS-0">
            <sch:assert test="width and not(width\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.WIDTH_TS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
```

© ISO 2007 – All rights reserved

```
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_TS">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_TS.DATE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_TS.DATE">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS.DATE" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_TS.DATE.FULL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE.FULL-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE.FULL-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATE.FULL-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
```


© ISO 2007 – All rights reserved

```
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_TS.DATE.FULL">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_TS.DATETIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_TS.DATETIME">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_TS.DATETIME.FULL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME.FULL-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME.FULL-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.DATETIME.FULL-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
    <xsd:restriction base="IVL_TS.DATETIME.FULL">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_TS.BIRTH">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.BIRTH-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.BIRTH-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_TS.BIRTH-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_TS.BIRTH">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_RTO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_RTO-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_RTO-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_RTO-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_RTO">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="0" />
```


© ISO 2007 – All rights reserved

```
        <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="RTO" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_MO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_MO-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_MO-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_MO-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_MO">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="MO" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_PQ">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_PQ">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="PQ" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_PQ.TIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ.TIME-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ.TIME-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_PQ.TIME-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_PQ.TIME">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_REAL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_REAL-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_REAL-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_REAL-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_REAL">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="REAL" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_INT">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_INT-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_INT-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_INT-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_INT">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="INT" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_INT.NONNEG">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="width">
        <sch:rule abstract="true" id="IVL.WIDTH_INT.NONNEG-0">
          <sch:assert test="width and not(width\@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="low">
        <sch:rule abstract="true" id="IVL.WIDTH_INT.NONNEG-1">
          <sch:assert test="not(low) and not(@lowClosed)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="high">
        <sch:rule abstract="true" id="IVL.WIDTH_INT.NONNEG-2">
          <sch:assert test="not(high) and not(@highClosed)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="IVL_INT.NONNEG">
      <xsd:sequence>
        <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
        <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
        <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
        <xsd:element name="any" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="IVL.WIDTH_INT.POS">
  <xsd:annotation>
    <xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
        <sch:pattern name="width">
          <sch:rule abstract="true" id="IVL.WIDTH_INT.POS-0">
            <sch:assert test="width and not(width\@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.WIDTH_INT.POS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.WIDTH_INT.POS-2">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT.POS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="any" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_CO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_CO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_CO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATE-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATE-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATE">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATE.FULL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATE.FULL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
```

© ISO 2007 – All rights reserved

```
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATETIME-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATETIME-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATETIME.FULL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS.DATETIME.FULL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_TS.BIRTH-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_TS.BIRTH-1">
```


© ISO 2007 – All rights reserved

```
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_RTO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_RTO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_RTO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_MO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_MO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_MO">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_PQ-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_PQ-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_PQ">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_PQ.TIME-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_PQ.TIME-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_REAL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
```


© ISO 2007 – All rights reserved

```
          <sch:rule abstract="true" id="IVL.HIGH_REAL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_REAL">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_INT-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_INT-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_INT.NONNEG-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_INT.NONNEG-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="IVL.HIGH_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="IVL.HIGH_INT.POS-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="IVL.HIGH_INT.POS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="IVL_INT.POS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="width" type="QTY" minOccurs="0" maxOccurs="0" />
          <xsd:element name="any" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_CO-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_CO-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
```


© ISO 2007 – All rights reserved

```
          <sch:rule abstract="true" id="SLIST_TS-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATE-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATE-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATE.FULL-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
```

© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATETIME-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_TS.DATETIME.FULL-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_TS.BIRTH-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_RTO-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_RTO-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_MO-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_MO-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_PQ">
    <xsd:annotation>
```

© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_PQ-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_PQ-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_PQ.TIME-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_PQ.TIME-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_REAL-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_REAL-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_INT-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_INT-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_INT.NONNEG-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SLIST_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="SLIST_INT.POS-0">
```

© ISO 2007 – All rights reserved

```
            <sch:assert test="@nullFlavor or (origin and not(origin\@nullFlavor) and
count(*[digits\not(@nullFlavor))&#62;0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on SLIST attributes">
          <sch:rule abstract="true" id="SLIST_INT.POS-1">
            <sch:assert test="count(*[self::origin or self::scale or
self::digits][@validTimeLow or @validTimeHigh or @controlActRoot or @controlActExtension or
@updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="origin" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="scale" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="digit" type="INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_CO-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_CO-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_CO-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_CO-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS-1">
```


© ISO 2007 – All rights reserved

```
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATE-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATE-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATE-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATE-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
```

© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATE.FULL-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATE.FULL-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATE.FULL-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME.FULL-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME.FULL-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS.DATETIME.FULL-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_TS.BIRTH-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_TS.BIRTH-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_TS.BIRTH-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
  </xsd:complexType>
  <xsd:complexType name="GLIST_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_RTO-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_RTO-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_RTO-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_RTO-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_MO-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_MO-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_MO-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_MO-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_PQ-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_PQ-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_PQ-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_PQ-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_PQ.TIME-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_PQ.TIME-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_PQ.TIME-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_PQ.TIME-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
```

© ISO 2007 – All rights reserved

```
        <xsd:sequence>
          <xsd:element name="head" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_REAL-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_REAL-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_REAL-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_REAL-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_INT-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_INT-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_INT-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_INT-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
```


© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_INT.NONNEG-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_INT.NONNEG-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_INT.NONNEG-3">
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GLIST_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="GLIST_INT.POS-0">
            <sch:assert test="@nullFlavor or (head and not(head\@nullFlavor) and increment and
not(increment\@nullFlavor) and @denominator)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="denominator must be positive">
          <sch:rule abstract="true" id="GLIST_INT.POS-1">
            <sch:assert test="not(@denominator) or (@denominator &#62; 0)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="period must be positive">
          <sch:rule abstract="true" id="GLIST_INT.POS-2">
            <sch:assert test="todo" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on GLIST attributes">
          <sch:rule abstract="true" id="GLIST_INT.POS-3">
```

© ISO 2007 – All rights reserved

```
            <sch:assert test="count(*[self::head or self::increment][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="head" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="increment" type="QTY" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="denominator" type="xsd:int" use="optional" />
        <xsd:attribute name="period" type="xsd:int" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_CO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_CO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS.DATE-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATE" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="second" type="QSET_TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATETIME.FULL" minOccurs="0" maxOccurs="1"
/>
          <xsd:element name="second" type="QSET_TS.DATETIME.FULL" minOccurs="0" maxOccurs="1"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_TS.BIRTH-0">
```

© ISO 2007 – All rights reserved

```
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_RTO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_MO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_MO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_PQ-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```


© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_PQ.TIME-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_REAL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_INT-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
```

© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSD_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSD_INT.POS-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.POS">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_CO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_CO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_CO">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_CO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="term" type="QSET_TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS.DATE-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS.DATE-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS.DATE.FULL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATE.FULL" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS.DATETIME-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="unbounded"
/>
```

© ISO 2007 – All rights reserved

```
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS.DATETIME.FULL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATETIME.FULL" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_TS.BIRTH-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_RTO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_RTO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_RTO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
```


© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_MO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_MO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_MO">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_MO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_PQ-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_PQ-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_PQ" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_PQ.TIME-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_PQ.TIME-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_REAL">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_REAL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_REAL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_REAL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_INT-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_INT-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSI_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSI_INT.NONNEG-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSI_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSI_INT.POS-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSI_INT.POS-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_INT.POS">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_INT.POS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_CO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_CO-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSU_CO-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_CO">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_CO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_TS">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_TS-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSU_TS-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_TS">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_TS" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_TS.DATE">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_TS.DATE-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
```

© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_TS.DATE-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_TS.DATE.FULL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATE.FULL" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_TS.DATETIME-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_TS.DATETIME.FULL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.DATETIME.FULL" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_TS.BIRTH-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_RTO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_RTO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_RTO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_MO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
```

© ISO 2007 – All rights reserved

```
        <sch:rule abstract="true" id="QSU_MO-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_MO">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_MO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_PQ">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_PQ-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSU_PQ-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_PQ">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_PQ" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_PQ.TIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_PQ.TIME-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSU_PQ.TIME-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_PQ.TIME">
      <xsd:sequence>
        <xsd:element name="term" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSU_REAL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSU_REAL-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSU_REAL-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
        </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_REAL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_INT-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_INT-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_INT.NONNEG-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSU_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSU_INT.POS-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSU_INT.POS-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 2)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.POS">
        <xsd:sequence>
          <xsd:element name="term" type="QSET_INT.POS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_CO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_CO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS.DATE-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.DATETIME.FULL" minOccurs="0" maxOccurs="1"
/>
          <xsd:element name="second" type="QSET_TS.DATETIME.FULL" minOccurs="0" maxOccurs="1"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
```

© ISO 2007 – All rights reserved

```
        <xsd:sequence>
          <xsd:element name="first" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_RTO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_MO-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_MO">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_PQ-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_PQ.TIME-0">
```


© ISO 2007 – All rights reserved

```
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_REAL-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_INT-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSP_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSP_INT.POS-0">
            <sch:assert test="@nullFlavor or (not(first/@nullFlavor) and
not(second/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.POS">
        <xsd:sequence>
          <xsd:element name="first" type="QSET_INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="second" type="QSET_INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_CO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_CO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_CO-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_CO-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_CO">
        <xsd:sequence>
          <xsd:element name="term" type="CO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="term" type="TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.DATE-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.DATE-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.DATE-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.DATE-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="term" type="TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.DATE.FULL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.DATE.FULL-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.DATE.FULL-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
```

© ISO 2007 – All rights reserved

```
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_TS.DATE.FULL">
      <xsd:sequence>
        <xsd:element name="term" type="TS.DATE.FULL" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_TS.DATETIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_TS.DATETIME-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_TS.DATETIME-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_TS.DATETIME-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_TS.DATETIME-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_TS.DATETIME">
      <xsd:sequence>
        <xsd:element name="term" type="TS.DATETIME" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_TS.DATETIME.FULL">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_TS.DATETIME.FULL-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_TS.DATETIME.FULL-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_TS.DATETIME.FULL-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_TS.DATETIME.FULL-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_TS.DATETIME.FULL">
      <xsd:sequence>
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="term" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.BIRTH-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_TS.BIRTH-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_TS.BIRTH-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="term" type="TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_RTO-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_RTO-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_RTO-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_RTO-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_RTO">
        <xsd:sequence>
          <xsd:element name="term" type="RTO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

© ISO 2007 – All rights reserved

```
<xsd:complexType name="QSS_MO">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_MO-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_MO-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_MO-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_MO-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_MO">
      <xsd:sequence>
        <xsd:element name="term" type="MO" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_PQ">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_PQ-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_PQ-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_PQ-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_PQ-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_PQ">
      <xsd:sequence>
        <xsd:element name="term" type="PQ" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_PQ.TIME">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_PQ.TIME-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_PQ.TIME-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_PQ.TIME-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_PQ.TIME-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="term" type="PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_REAL-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_REAL-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_REAL-2">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_REAL-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_REAL">
        <xsd:sequence>
          <xsd:element name="term" type="REAL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSS_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="not null">
          <sch:rule abstract="true" id="QSS_INT-0">
            <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_INT-1">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
```

© ISO 2007 – All rights reserved

```
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_INT-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_INT-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_INT">
      <xsd:sequence>
        <xsd:element name="term" type="INT" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_INT.NONNEG">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_INT.NONNEG-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_INT.NONNEG-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_INT.NONNEG-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_INT.NONNEG-3">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="QSET_INT.NONNEG">
      <xsd:sequence>
        <xsd:element name="term" type="INT.NONNEG" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QSS_INT.POS">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_INT.POS-0">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="size">
        <sch:rule abstract="true" id="QSS_INT.POS-1">
          <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
        </sch:rule>
      </sch:pattern>
      <sch:pattern name="not null">
        <sch:rule abstract="true" id="QSS_INT.POS-2">
          <sch:assert test="@nullFlavor or not(terms/@nullFlavor)" />
        </sch:rule>
      </sch:pattern>
```


© ISO 2007 – All rights reserved

```
        <sch:pattern name="size">
          <sch:rule abstract="true" id="QSS_INT.POS-3">
            <sch:assert test="@nullFlavor or (count(*[self::terms]) &#62;= 1)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_INT.POS">
        <xsd:sequence>
          <xsd:element name="term" type="INT.POS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS.DATE-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS.DATE-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS.DATE.FULL-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS.DATE.FULL-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS.DATETIME-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS.DATETIME-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS.DATETIME.FULL-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS.DATETIME.FULL-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="PIVL_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="period or frequency">
          <sch:rule abstract="true" id="PIVL_TS.BIRTH-0">
            <sch:assert test="not(period) or not(frequency)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on PIVL attributes">
          <sch:rule abstract="true" id="PIVL_TS.BIRTH-1">
            <sch:assert test="count(*[self::phase or self::period][@validTimeLow or
@validTimeHigh or @controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="phase" type="IVL_TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="period" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="frequency" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="count" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="alignment" type="CalendarCycle" use="optional" />
        <xsd:attribute name="isFlexible" type="xsd:boolean" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS.DATE">
    <xsd:annotation>
```

© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATE-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATE-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATE.FULL-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATETIME-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS.DATETIME.FULL-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="EIVL_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="required attributes">
          <sch:rule abstract="true" id="EIVL_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or (event)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="no updateMode or History on EIVL attributes">
          <sch:rule abstract="true" id="EIVL_TS.BIRTH-1">
            <sch:assert test="count(*[self::offset][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QSET_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="offset" type="IVL_PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="event" type="TimingEvent" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_BL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_BL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_BL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_BL">
    <xsd:annotation>
      <xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_BL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_BL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_BL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="BL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_BL.NONNULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_BL.NONNULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_BL.NONNULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_BL.NONNULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_BL.NONNULL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_BL.NONNULL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_BL.NONNULL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="BL.NONNULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.IMAGE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.IMAGE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.IMAGE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.IMAGE">
    <xsd:annotation>
```

© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.IMAGE-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.IMAGE-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.IMAGE-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.IMAGE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.TEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.TEXT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.TEXT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.TEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.TEXT-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.TEXT-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.TEXT-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.DOC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.DOC-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.DOC" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.DOC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.DOC-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.DOC-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.DOC-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.DOC" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.DOC.REF">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.DOC.REF-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.DOC.REF" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.DOC.REF">
```

© ISO 2007 – All rights reserved

```
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.DOC.REF-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.DOC.REF-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.DOC.REF-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.DOC.REF" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.DOC.INLINE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.DOC.INLINE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.DOC.INLINE" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.DOC.INLINE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.DOC.INLINE-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.DOC.INLINE-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.DOC.INLINE-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="value" type="ED.DOC.INLINE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.SIGNATURE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.SIGNATURE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.SIGNATURE" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.SIGNATURE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.SIGNATURE-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.SIGNATURE-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.SIGNATURE-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.SIGNATURE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ED.STRUCTUREDTEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ED.STRUCTUREDTEXT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ED.STRUCTUREDTEXT" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
```

© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ED.STRUCTUREDTEXT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ED.STRUCTUREDTEXT-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ED.STRUCTUREDTEXT-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ED.STRUCTUREDTEXT-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ED.STRUCTUREDTEXT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ST">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ST-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ST" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ST">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ST-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ST-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ST-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ST" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_ST.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_ST.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_ST.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_ST.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_ST.NT-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_ST.NT-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_ST.NT-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="ST.NT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_SC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_SC-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_SC" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
```

© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_SC">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_SC-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_SC-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_SC-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="SC" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_SC.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_SC.NT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_SC.NT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_SC.NT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_SC.NT-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_SC.NT-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_SC.NT-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
```


© ISO 2007 – All rights reserved

```
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="SC.NT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TEL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TEL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TEL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TEL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TEL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TEL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TEL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TEL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TEL.EMAIL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TEL.EMAIL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TEL.EMAIL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
```

© ISO 2007 – All rights reserved

```
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TEL.EMAIL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TEL.EMAIL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TEL.EMAIL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TEL.EMAIL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TEL.EMAIL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TEL.PERSON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TEL.PERSON-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TEL.PERSON" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TEL.PERSON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TEL.PERSON-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TEL.PERSON-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TEL.PERSON-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TEL.PERSON" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TEL.PHONE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TEL.PHONE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TEL.PHONE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TEL.PHONE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TEL.PHONE-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TEL.PHONE-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TEL.PHONE-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TEL.PHONE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TEL.URL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TEL.URL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TEL.URL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TEL.URL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TEL.URL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TEL.URL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TEL.URL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TEL.URL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_II">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_II-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_II" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_II">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_II-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_II-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_II-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="II" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_CD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_CD-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_CD" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_CD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_CD-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_CD-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_CD-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="CD" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_CD.CV">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_CD.CV-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_CD.CV" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_CD.CV">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_CD.CV-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_CD.CV-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_CD.CV-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="CD.CV" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_CO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_CO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_CO-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_CO-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_CO-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_CS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_CS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_CS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_CS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_CS-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_CS-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_CS-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="CS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_AD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_AD-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_AD" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_AD">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_AD-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_AD-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_AD-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="AD" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_EN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_EN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_EN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_EN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_EN-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_EN-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_EN-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="EN" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_EN.TN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_EN.TN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_EN.TN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_EN.TN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_EN.TN-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_EN.TN-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_EN.TN-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="EN.TN" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_EN.PN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_EN.PN-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_EN.PN" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_EN.PN">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_EN.PN-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_EN.PN-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_EN.PN-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="EN.PN" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_EN.ON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_EN.ON-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_EN.ON" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_EN.ON">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_EN.ON-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_EN.ON-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_EN.ON-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="EN.ON" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS.DATE-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TS.DATE" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
```

© ISO 2007 – All rights reserved

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS.DATE-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS.DATE-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS.DATE-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS.DATE.FULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TS.DATE.FULL" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS.DATE.FULL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS.DATE.FULL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS.DATE.FULL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
```


© ISO 2007 – All rights reserved

```
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS.DATETIME-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TS.DATETIME" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS.DATETIME-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS.DATETIME-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS.DATETIME-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS.DATETIME.FULL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="item" type="UVP_TS.DATETIME.FULL" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS.DATETIME.FULL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS.DATETIME.FULL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS.DATETIME.FULL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_TS.BIRTH-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_TS.BIRTH" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_TS.BIRTH-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_TS.BIRTH-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_TS.BIRTH-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_RTO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_RTO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_RTO-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_RTO-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_RTO-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_MO-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
```

© ISO 2007 – All rights reserved

```
        <xsd:sequence>
          <xsd:element name="item" type="UVP_MO" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_MO-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_MO-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_MO-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_PQ-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_PQ" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_PQ-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_PQ-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_PQ-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_PQ.TIME-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_PQ.TIME" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_PQ.TIME-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_PQ.TIME-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_PQ.TIME-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_REAL-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
```

© ISO 2007 – All rights reserved

```
        <xsd:sequence>
          <xsd:element name="item" type="UVP_REAL" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_REAL-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_REAL-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_REAL-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_INT-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_INT" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_INT-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_INT-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_INT-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_INT.NONNEG-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_INT.NONNEG" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_INT.NONNEG-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_INT.NONNEG-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_INT.NONNEG-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="NPPD_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="must have at least one item">
          <sch:rule abstract="true" id="NPPD_INT.POS-0">
            <sch:assert test="(@nullFlavor or item) and not(@nullFlavor and item)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="item" type="UVP_INT.POS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="UVP_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="value not null">
          <sch:rule abstract="true" id="UVP_INT.POS-0">
            <sch:assert test="@nullFlavor or (value and not(value/@nullFlavor))" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="must have probability">
          <sch:rule abstract="true" id="UVP_INT.POS-1">
            <sch:assert test="@nullFlavor or @probability" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="No History or Update Mode">
          <sch:rule abstract="true" id="UVP_INT.POS-2">
            <sch:assert test="count(*[self::value][@validTimeLow or @validTimeHigh or
@controlActRoot or @controlActExtension or @updateMode])=0" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ANY">
        <xsd:sequence>
          <xsd:element name="value" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="probability" type="xsd:double" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="QSET.BOUNDEDPIVL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="QSET.BOUNDEDPIVL 2">
          <sch:rule abstract="true" id="QSET.BOUNDEDPIVL-0">
            <sch:assert test="count(*)=2" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="QSI_TS">
        <xsd:sequence>
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="term" type="QSET_TS" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_BL">
    <xsd:complexContent>
      <xsd:extension base="LIST_BL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_BL.NONNULL">
    <xsd:complexContent>
      <xsd:extension base="LIST_BL.NONNULL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.IMAGE">
```


© ISO 2007 – All rights reserved

```
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.IMAGE" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.TEXT">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.TEXT" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.DOC">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.DOC" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.DOC.REF">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.DOC.REF" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.DOC.INLINE">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.DOC.INLINE" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.SIGNATURE">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.SIGNATURE" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.STRUCTUREDTEXT">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.STRUCTUREDTEXT" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ED.STRUCTUREDTITLE">
    <xsd:complexContent>
      <xsd:extension base="LIST_ED.STRUCTUREDTITLE" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ST">
    <xsd:complexContent>
      <xsd:extension base="LIST_ST" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_ST.NT">
    <xsd:complexContent>
      <xsd:extension base="LIST_ST.NT" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_SC">
    <xsd:complexContent>
      <xsd:extension base="LIST_SC" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_SC.NT">
    <xsd:complexContent>
      <xsd:extension base="LIST_SC.NT" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TEL">
    <xsd:complexContent>
      <xsd:extension base="LIST_TEL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TEL.EMAIL">
    <xsd:complexContent>
      <xsd:extension base="LIST_TEL.EMAIL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TEL.PERSON">
    <xsd:complexContent>
      <xsd:extension base="LIST_TEL.PERSON" />
```

© ISO 2007 – All rights reserved

```
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_TEL.PHONE">
  <xsd:complexContent>
    <xsd:extension base="LIST_TEL.PHONE" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_TEL.URL">
  <xsd:complexContent>
    <xsd:extension base="LIST_TEL.URL" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_II">
  <xsd:complexContent>
    <xsd:extension base="LIST_II" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_CD">
  <xsd:complexContent>
    <xsd:extension base="LIST_CD" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_CD.CV">
  <xsd:complexContent>
    <xsd:extension base="LIST_CD.CV" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_CO">
  <xsd:complexContent>
    <xsd:extension base="LIST_CO" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_CS">
  <xsd:complexContent>
    <xsd:extension base="LIST_CS" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_AD">
  <xsd:complexContent>
    <xsd:extension base="LIST_AD" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_EN">
  <xsd:complexContent>
    <xsd:extension base="LIST_EN" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_EN.TN">
  <xsd:complexContent>
    <xsd:extension base="LIST_EN.TN" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_EN.PN">
  <xsd:complexContent>
    <xsd:extension base="LIST_EN.PN" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_EN.ON">
  <xsd:complexContent>
    <xsd:extension base="LIST_EN.ON" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_TS">
  <xsd:complexContent>
    <xsd:extension base="LIST_TS" />
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="HIST_TS.DATE">
  <xsd:complexContent>
    <xsd:extension base="LIST_TS.DATE" />
  </xsd:complexContent>
</xsd:complexType>
```


© ISO 2007 – All rights reserved

```
  <xsd:complexType name="HIST_TS.DATE.FULL">
    <xsd:complexContent>
      <xsd:extension base="LIST_TS.DATE.FULL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TS.DATETIME">
    <xsd:complexContent>
      <xsd:extension base="LIST_TS.DATETIME" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TS.DATETIME.FULL">
    <xsd:complexContent>
      <xsd:extension base="LIST_TS.DATETIME.FULL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_TS.BIRTH">
    <xsd:complexContent>
      <xsd:extension base="LIST_TS.BIRTH" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_RTO">
    <xsd:complexContent>
      <xsd:extension base="LIST_RTO" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_MO">
    <xsd:complexContent>
      <xsd:extension base="LIST_MO" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_PQ">
    <xsd:complexContent>
      <xsd:extension base="LIST_PQ" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_PQ.TIME">
    <xsd:complexContent>
      <xsd:extension base="LIST_PQ.TIME" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_REAL">
    <xsd:complexContent>
      <xsd:extension base="LIST_REAL" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_INT">
    <xsd:complexContent>
      <xsd:extension base="LIST_INT" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_INT.NONNEG">
    <xsd:complexContent>
      <xsd:extension base="LIST_INT.NONNEG" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="HIST_INT.POS">
    <xsd:complexContent>
      <xsd:extension base="LIST_INT.POS" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_CO-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
```

© ISO 2007 – All rights reserved

```
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS.DATE-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS.DATE.FULL-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
```


© ISO 2007 – All rights reserved

```
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS.DATETIME-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS.DATETIME.FULL-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_TS.BIRTH-0">
```

© ISO 2007 – All rights reserved

```
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_RTO-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_MO-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_PQ">
    <xsd:annotation>
```


© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_PQ-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_PQ.TIME-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_REAL-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
```

© ISO 2007 – All rights reserved

```
  </xsd:complexType>
  <xsd:complexType name="URG_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_INT-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_INT.NONNEG-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="no inner attributes">
          <sch:rule abstract="true" id="URG_INT.POS-0">
            <sch:assert test="not(@validTimeLow) and not(@validTimeHigh) and not
(@controlActRoot) and not(@controlActExtension) and not(@flavorId) and not(@nullFlavor) and
not(@updateMode) and not(expression) and not(originalText) and not(uncertainty) and
not(@uncertaintyType)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="QTY">
        <xsd:sequence>
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="optional" />
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_CO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_CO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_CO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS.DATE-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
```

© ISO 2007 – All rights reserved

```
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS.DATE-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATE">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS.DATE.FULL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS.DATE.FULL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS.DATETIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS.DATETIME-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS.DATETIME-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS.DATETIME.FULL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS.DATETIME.FULL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_TS.BIRTH-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_TS.BIRTH-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_RTO">
    <xsd:annotation>
```

© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_RTO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_RTO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_RTO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_MO-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_MO-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_MO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_PQ-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_PQ-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
```


© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_PQ">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_PQ.TIME-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_PQ.TIME-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_REAL-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_REAL-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_REAL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
```

© ISO 2007 – All rights reserved

```
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_INT-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_INT-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_INT.NONNEG-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_INT.NONNEG-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.LOW_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.LOW_INT.POS-0">
            <sch:assert test="low and not(low\@nullFlavor) and (@lowClosed = &#34;true&#34;)"
/>
          </sch:rule>
```


© ISO 2007 – All rights reserved

```
        </sch:pattern>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.LOW_INT.POS-1">
            <sch:assert test="not(high) and not(@highClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT.POS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="1" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="highClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_CO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_CO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_CO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_CO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="CO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="CO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
```

© ISO 2007 – All rights reserved

```
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS.DATE">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATE-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATE-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATE">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATE" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS.DATE.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATE.FULL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATE.FULL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATE.FULL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATE.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATE.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS.DATETIME">
    <xsd:annotation>
```


© ISO 2007 – All rights reserved

```
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATETIME-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATETIME-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATETIME">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATETIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS.DATETIME.FULL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATETIME.FULL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS.DATETIME.FULL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.DATETIME.FULL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.DATETIME.FULL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_TS.BIRTH">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_TS.BIRTH-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_TS.BIRTH-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
```

© ISO 2007 – All rights reserved

```
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_TS.BIRTH">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="TS.BIRTH" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="TS.BIRTH" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_RTO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_RTO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_RTO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_RTO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="RTO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="RTO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_MO">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_MO-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_MO-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_MO">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="MO" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="MO" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
```


© ISO 2007 – All rights reserved

```
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_PQ">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_PQ-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_PQ-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_PQ">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="PQ" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_PQ.TIME">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_PQ.TIME-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_PQ.TIME-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_PQ.TIME">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="PQ.TIME" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="PQ.TIME" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_REAL">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_REAL-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
```

© ISO 2007 – All rights reserved

```
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_REAL-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_REAL">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="REAL" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="REAL" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_INT">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_INT-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_INT-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_INT.NONNEG">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_INT.NONNEG-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_INT.NONNEG-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT.NONNEG">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
```


© ISO 2007 – All rights reserved

```
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.NONNEG" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT.NONNEG" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="URG.HIGH_INT.POS">
    <xsd:annotation>
      <xsd:appinfo>
        <sch:pattern name="high">
          <sch:rule abstract="true" id="URG.HIGH_INT.POS-0">
            <sch:assert test="high and not(high\@nullFlavor) and (@highClosed =
&#34;true&#34;)" />
          </sch:rule>
        </sch:pattern>
        <sch:pattern name="low">
          <sch:rule abstract="true" id="URG.HIGH_INT.POS-1">
            <sch:assert test="not(low) and not(@lowClosed)" />
          </sch:rule>
        </sch:pattern>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:restriction base="URG_INT.POS">
        <xsd:sequence>
          <xsd:element name="expression" type="ED" minOccurs="0" maxOccurs="1" />
          <xsd:element name="originalText" type="ED.TEXT" minOccurs="0" maxOccurs="1" />
          <xsd:element name="uncertainty" type="QTY" minOccurs="0" maxOccurs="1" />
          <xsd:element name="low" type="INT.POS" minOccurs="0" maxOccurs="0" />
          <xsd:element name="high" type="INT.POS" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="nullFlavor" type="NullFlavor" fixed="UNK" />
        <xsd:attribute name="lowClosed" type="xsd:boolean" use="prohibited" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

© ISO 2007 – All rights reserved