



1

2

3

4

5

**Virtual Medical Record (vMR) for Clinical Decision  
Support – GELLO Implementation Guide  
(V3IG\_CDS\_VMR\_GELLO\_R1\_I1\_2010MAY)**

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

**Document Editor  
Andrew McIntyre MBBS, FRACP; Medical-Objects**

**Kensaku Kawamoto, MD, PhD; Duke University  
Guilherme Del Fiol, MD, PhD; Duke University  
Peter R. Tattam; Tattam Software Enterprises Pty Ltd  
Peter Scott, MBBS; Medical-Objects**

**Project Sponsor  
HL7 Clinical Decision Support**

**HL7 Project #184  
Informative Ballot  
May 2010**

25 **Table of Contents**

---

26 *Table of Contents* ..... 2

27 *1 Executive Summary*..... 3

28 *2 GELLO implementation guide for VMR*..... 3

29

30

## 31 1 Executive Summary

---

32 A Virtual Medical Record (vMR) for Clinical Decision Support (CDS) is a data model  
33 for representing clinical information *inputs* and *outputs* that can be exchanged  
34 between CDS engines and local clinical information systems, through mechanisms  
35 such as CDS services. A vMR for CDS is needed to enable the design and  
36 development of scalable CDS resources that can be used across multiple healthcare  
37 institutions and health information systems.

38  
39 The objective of the HL7 CDS Work Group's vMR GELLO Implementation Guide is to  
40 allow tight integration of the VMR structure with a GELLO compiler, ensuring good  
41 interoperability of CDS logic between institutions and fostering reliable type  
42 checking and code insight by GELLO implementations. This guide describes a  
43 grammar for defining GELLO classes and data types. It also contains draft  
44 definitions of the base GELLO types and ISO 21090 data types. The final VMR would  
45 also be encoded in this format allowing compilers access to the final VMR model.  
46 This format will be identical to the semantics of a UML XMI representation, but is  
47 compiler friendly and more human readable. The ISO 21090 data types are an  
48 implementation of the HL7 V3 R2 data types.

49  
50 This is the first informative ballot for this material. Input is sought on the format. It  
51 is envisaged that all elements of the final VMR, when this is finalised, will be  
52 represented in both UML (.xmi files) and this format.

53  
54

## 55 2 GELLO implementation Guide for VMR

56

### 57 2.1 Grammar for GELLO class definitions

58

59 This is the GELLO class BNF which uses the Extended BNF syntax as used in the GELLO  
60 standard.

61

```
GelloPackage ::= PackageStatement
```

62

```
PackageStatement ::=
```

63

```
<PACKAGE> NameOrLiteral ImportsStatement? PackageElement* <ENDPACKAGE>
```

64

65

```
ImportsStatement ::=
```

66

```
<IMPORTS> ImportName ("," ImportName)*
```

67

```
ImportName ::= NameOrLiteral
```

68

69

```
PackageElement ::= PackageStatement | ClassDefinition
```

70

71

```
ClassDefinition ::=
```

72

```
<CLASS> ClassDefName (<EXTENDS> ClassRef)? ClassMember* |
```

73

```
<CLASS> ClassDefName <EQUAL> ClassRef |
```

74

```
<CLASS> ClassName EnumerationType
```

75

```
ClassDefName ::=
```

76

```
NameOrLiteral |
```

77

```
NameOrLiteral "(" ClassDefParams ")"
```

78

```
ClassDefParams ::= ClassRefOrParam ("," ClassRefOrParam)*
```

79

```
ClassRefOrParam ::= ClassRef |
```

80

```
Name ":" ClassRef
```

81

```
ClassRef ::=
```

82

```
ClassName |
```

83

```
ClassName "(" ClassRef ")" |
```

84

```
<TUPLE>
```

85

```
ClassMember ::= NameOrLiteral ( "(" ClassFormalParams? ")" )? ":"
```

86

```
ClassFormalParams ::= ClassFormalParam ("," ClassFormalParam )*
```

87

```
ClassFormalParam ::= Name ":" ClassRef
```

88

```
EnumerationType ::= <ENUM> "(" (EnumLiteral ( "," EnumLiteral )*) ")"
```

89

```
EnumLiteral ::= NameOrLiteral
```

90

```
NameOrLiteral ::= Name | <STRING_LITERAL>
```

91

```
ClassName ::= NameWithPath | <STRING_LITERAL>
```

92

```
NameWithPath ::= Name ( "::" Name )*
```

93

```
Name ::= <ID> ( "." <ID> )*
```

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

```

112 <STRING_LITERAL: ('\'\' (~[ \'\' , "\n", "\r"])* \'\' | "\'\' (~[ \'\' , "\n",
113 "\r"])* \'\' ) >
114
115 <ID: ["a"- "z", "A"- "Z"] (["a"- "z", "A"- "Z", "0"- "9"] | "_"(["a"- "z", "A"- "Z", "0"-
116 "9"])+)* >
117
118 <ENUM: "Enum">
119
120 <TUPLE: "Tuple" >
121
122 <EQUAL: "=" >
123
124 <ENDPACKAGE: "endPackage" | "EndPackage" | "endpackage">
125
126 <CLASS: "class"| "Class">
127
128 <EXTENDS: "extends" >
129
130 <IMPORTS: "Imports" | "imports">
131
132 <PACKAGE: "Package" | "package">
133
134

```

## 2.2 GELLO base types class definition

These are the base GELLO types which are enhanced OCL types. They are used by the ISO 21090 datatypes.

```

135
136
137
138
139
140 Package System
141
142 Class Any
143     "="(o: Any): Boolean
144     "<>"(o: Any): Boolean
145     oclIsDefined(): Boolean
146     oclAsType(t: T): T
147     oclIsTypeOf(t: T): Boolean
148     oclIsKindOf(typename: String): Boolean
149     oclIsInState(statename: String): Boolean
150     allInstances(): Set(T)
151
152 Class Real extends Any
153     "-"(): Real
154     "+"(rhs: Real): Real
155     "-"(rhs: Real): Real
156     "*" (rhs: Real): Real
157     "/"(rhs: Real): Real
158     abs(): Real
159     floor(): Integer
160     round(): Integer
161     "max"(rhs: Real): Real
162     "min"(rhs: Real): Real
163     "="(rhs: Real): Boolean
164     "<>"(rhs: Real): Boolean
165     "<"(rhs: Real): Boolean
166     "<="(rhs: Real): Boolean
167     ">"(rhs: Real): Boolean
168     ">="(rhs: Real): Boolean
169     toChar():String
170     toCharFormat(decimalplaces:Integer): String
171
172 Class Integer extends Real
173     "-"(): Integer

```

```

174     "+"(rhs: Integer): Integer
175     "-"(rhs: Integer): Integer
176     "*" (rhs: Integer): Integer
177     "div"(rhs: Integer): Integer
178     "mod"(rhs: Integer): Integer
179     "max"(rhs: Integer): Integer
180     "min"(rhs: Integer): Integer
181     abs(): Integer
182     toChar():String
183
184 Class String extends Any
185     "="(rhs: String): Boolean
186     "<>"(rhs: String): Boolean
187     size(): Integer
188     concat(rhs: String): String
189     substring(lower:Integer, upper: Integer): String
190     toInteger(): Integer
191     toReal(): Real
192     toUpper(): String
193     toLower():String
194     lpad(size:Integer, padChar:String):String
195     rpad(size:Integer, padChar:String):String
196     ltrim(trimChars:String):String
197     rtrim(trimChars:String):String
198     replace(replaceThis:String, withThis:String):String
199
200 Class Boolean extends Any
201     "="(rhs: Boolean): Boolean
202     "<>"(rhs: Boolean): Boolean
203     "or"(rhs: Boolean): Boolean
204     "xor"(rhs: Boolean): Boolean
205     "and"(rhs: Boolean): Boolean
206     "not"(): Boolean
207     implies(rhs: Boolean): Boolean
208
209 Class Collection extends Any
210
211     count(o: T): Integer
212     excludes(o: T): Boolean
213     excludesAll(c: Collection): Boolean
214     includes(o: T): Boolean
215     includesAll(c: Collection): Boolean
216     isEmpty(): Boolean
217     notEmpty(): Boolean
218     size(): Integer
219     sum(): T
220     "max"(): T
221     "min"(): T
222
223     "="(s: Collection): Boolean
224     "<>"(s: Collection): Boolean
225
226     asSet(): Set(T)
227     asBag(): Bag(T)
228     asSequence(): Sequence(T)
229
230     exists(e: Boolean): Boolean
231     forAll(e: Any): Boolean
232     collect(e: Any): Bag(T)
233     sortedBy(e: Any): Sequence(T)
234     Average():Real
235     Stdev():Real
236     Variance():Real
237     Median():Real
238     Mode():Real
239     Join(): Collection(Tuple)

```

```

240     Like(): Collection(String)
241     NotLike(): Collection(String)
242     Between(string1: String, String2: String): Collection(String)
243     Distinct(): Set(T)
244
245
246     Class "Set" extends Collection
247     "-"(s: Set(T)): Set(T)
248     union(c: Bag(T)): Bag(T)
249     union(s: Set(T)): Set(T)
250     intersection(b: Bag(T)): Bag(T)
251     intersection(s: Set(T)): Set(T)
252     including(o: T): Set(T)
253     excluding(o: T): Set(T)
254     symmetricDifference(s: Set(T)): Set(T)
255     flatten(): Set(T)
256
257     select(e: Boolean): Set(T)
258     reject(e: Boolean): Set(T)
259
260
261     Class "Bag" extends Collection
262     union(b: Bag(T)): Bag(T)
263     union(s: Set(T)): Bag(T)
264     intersection(b: Bag(T)): Bag(T)
265     intersection(s: Set(T)): Bag(T)
266     including(o: T): Bag(T)
267     excluding(o: T): Bag(T)
268     flatten(): Bag(T)
269
270     select(e: Boolean): Bag(T)
271     reject(e: Boolean): Bag(T)
272
273     Class "Sequence" extends Collection
274     union(s: Sequence(T)): Sequence(T)
275     including(o: T): Sequence(T)
276     excluding(o: T): Sequence(T)
277
278     append(o: T): Sequence(T)
279     at(i: Integer): T
280     first(): T
281     flatten(): Sequence(T)
282     indexOf(o: T): Integer
283     insertAt(i: Integer, o: T): Sequence(T)
284     last(): T
285     prepend(o: T): Sequence(T)
286     subSequence(lower: Integer, upper: Integer): Sequence(T)
287
288     select(e: Boolean): Sequence(T)
289     reject(e: Boolean): Sequence(T)
290     lastN(number: Integer): Sequence(T)
291     firstN(number: Integer): Sequence(T)
292     Reverse(): Sequence(T)
293
294     EndPackage
295

```

## 2.3 ISO2190 Data Types Definition

296  
297  
298 These data types are an implementation of the HL7 V3 Release 2 data types which are  
299 likely to be used for the final VMR model. They were originally imported from the xml  
300 schema file but also define operations.  
301

```

302 Package iso_21090_datatypes -- Imported iso-21090-datatypes.xsd 17/12/2009
303 3:12:39 PM
304
305
306 class HXIT
307     controlInformationExtension: String
308     controlInformationRoot: Uid
309     validTimeHigh: String
310     validTimeLow: String
311
312 class NullFlavor Enum("NI", "INV", "OTH", "NINF", "PINF", "UNC", "DER",
313 "UNK", "ASKU", "NAV", "NASK", "QS", "TRC", "MSK", "NA")
314
315 class UpdateMode Enum("A", "D", "R", "AR", "N", "U", "K")
316
317 class ANY extends HXIT
318     flavorId: String
319     nullFlavor: NullFlavor
320     updateMode: UpdateMode
321     --operations
322     equals(other : ANY):BL
323
324 class XP
325     code: String
326     codeSystem: String
327     codeSystemVersion: String
328     language: Code
329     nullFlavor: NullFlavor
330     value: String
331
332 class AddressPartType Enum("AL", "ADL", "UNID", "UNIT", "DAL", "DINST",
333 "DINSTA", "DINSTQ", "DMOD", "DMODID", "SAL", "BNR", "BNN", "BNS", "STR",
334 "STB", "STTYP", "DIR", "INT", "CAR", "CEN", "CNT", "CPA", "CTY", "DEL", "POB",
335 "PRE", "STA", "ZIP", "DPID")
336
337 class ADXP extends XP
338     type: AddressPartType
339
340 class PostalAddressUse Enum("H", "HP", "HV", "WP", "DIR", "PUB", "BAD",
341 "PHYS", "PST", "TMP", "ABC", "IDE", "SYL", "SRCH", "SNDX", "PHON")
342
343 class AD extends ANY
344     isNotOrdered: Boolean
345     part: Sequence(ADXP)
346     use: Set(PostalAddressUse)
347     useablePeriod: QSET_TS
348
349 class BL extends ANY
350     value: Boolean
351     --operations
352     "and"(other : BL) : BL
353     "or"(other : BL) : BL
354     "xor"(other : BL) : BL
355     implies(other : BL) : BL
356     "not"() : BL
357
358 class CodingRationale Enum("O", "P", "R", "OR", "PR")
359
360 class CD extends ANY
361     code: String
362     codeSystem: Uid
363     codeSystemName: String
364     codeSystemVersion: String
365     codingRationale: CodingRationale
366     displayName: ST
367     id: String

```

```

368     originalText: ED
369     source: XReference
370     translation: Sequence(CD)
371     valueSet: Uid
372     valueSetVersion: String
373     --operations
374     --implies(other : ANY) : BL
375     implies(other : CS) : BL
376     implies(other : CD) : BL
377
378
379     class CO extends QTY
380     code: CD
381     value: Decimal
382     --operations
383     "+"(other : CO) : CO
384     "-"(other : CO) : CO
385     "min"(other : CO) : CO
386     "max"(other : CO) : CO
387
388     class CS extends ANY
389     code: String
390     --operations
391     codeSystem : Uid
392     codeSystemName : String
393     codeSystemVersion : String
394     --implies(other : Any): BL
395     implies(other : CS) : BL
396     implies(other : CD) : BL
397
398     class Decimal extends Real
399
400     class Compression Enum("DF", "GZ", "ZL", "Z", "BZ", "Z7")
401
402     class Code extends String
403
404     class IntegrityCheckAlgorithm Enum("SHA1", "SHA256")
405
406     class ED extends ANY
407     charset: Code
408     compression: Compression
409     data: String
410     description: ST
411     integrityCheck: String
412     integrityCheckAlgorithm: IntegrityCheckAlgorithm
413     language: Code
414     mediaType: String
415     reference: TEL
416     thumbnail: ED
417     translation: Sequence(ED)
418     value: String
419     xml: Any
420     --operations
421     --<TODO: Chase up Binary in gello context?>
422     --canonical : Binary
423
424     class TimingEvent Enum("HS", "WAKE", "AC", "ACM", "ACD", "ACV", "IC", "ICM",
425 "ICD", "ICV", "PC", "PCM", "PCD", "PCV", "C", "CM", "CD", "CV")
426
427     class EIVL_TS extends QSET_TS
428     event: TimingEvent
429     offset: IVL_PQ
430
431     class EntityNamePartQualifier Enum("LS", "AC", "NB", "PR", "HON", "BR",
432 "AD", "SP", "MID", "CL", "IN", "PFX", "SFX")
433

```

```

434     class EntityNamePartType Enum("FAM", "GIV", "TITLE", "DEL")
435
436     class ENXP extends XP
437         qualifier: Set(EntityNamePartQualifier)
438         type: EntityNamePartType
439
440     class EntityNameUse Enum("ABC", "SYL", "IDE", "C", "OR", "T", "I", "P",
441 "ANON", "A", "R", "OLD", "DN", "M", "SRCH", "PHON")
442
443     class EN extends ANY
444         part: Sequence(ENXP)
445         use: set_EntityNameUse
446         --operations
447         canonical() : EN
448
449     class GLIST_PQ extends ANY
450         denominator: Integer
451         head: PQ
452         increment: QTY
453         period: Integer
454
455     class GLIST_REAL extends ANY
456         denominator: Integer
457         head: REAL
458         increment: QTY
459         period: Integer
460
461     class GLIST_TS extends ANY
462         denominator: Integer
463         head: TS
464         increment: QTY
465         period: Integer
466
467     class IdentifierReliability Enum("ISS", "VRF", "UNV")
468
469     class IdentifierScope Enum("BUSN", "OBJ", "VER", "VW")
470
471     class II extends ANY
472         displayable: Boolean
473         extension: String
474         identifierName: String
475         reliability: IdentifierReliability
476         root: Uid
477         scope: IdentifierScope
478
479     class INT extends QTY
480         value: Integer
481         --operations
482         "+"(other : INT) : INT
483         "-"(other : INT) : INT
484         "*" (other : INT) : INT
485         "/"(other : INT) : REAL
486         "/"(other : REAL) : REAL
487         abs() : INT
488         "div"(other : INT) : INT
489         "mod"(other : INT) : INT
490         "min"(other : INT) : INT
491         "max"(other : INT) : INT
492
493     class IVL_CO extends QSET_CO
494         any: CO
495         high: CO
496         highClosed: Boolean
497         low: CO
498         lowClosed: Boolean
499         width: QTY

```

```

500
501 class IVL_INT extends QSET_INT
502     any: INT
503     high: INT
504     highClosed: Boolean
505     low: INT
506     lowClosed: Boolean
507     width: QTY
508
509 class IVL_MO extends QSET_MO
510     any: MO
511     high: MO
512     highClosed: Boolean
513     low: MO
514     lowClosed: Boolean
515     width: QTY
516
517 class IVL_PQ extends QSET_PQ
518     any: PQ
519     high: PQ
520     highClosed: Boolean
521     low: PQ
522     lowClosed: Boolean
523     width: QTY
524
525 class IVL_QTY extends QSET_QTY
526     any: QTY
527     high: QTY
528     highClosed: Boolean
529     low: QTY
530     lowClosed: Boolean
531     width: QTY
532
533 class IVL_REAL extends QSET_REAL
534     any: REAL
535     high: REAL
536     highClosed: Boolean
537     low: REAL
538     lowClosed: Boolean
539     width: QTY
540
541 class IVL_TS extends QSET_TS
542     any: TS
543     high: TS
544     highClosed: Boolean
545     low: TS
546     lowClosed: Boolean
547     width: QTY
548
549 class MO extends QTY
550     currency: Code
551     value: Decimal
552     --operations
553     "+"(other : MO) : MO
554     "-"(other : MO) : MO
555     "*" (other : REAL) : MO
556     "/"(other : REAL) : MO
557     "min"(other : MO) : MO
558     "max"(other : MO) : MO
559
560 class CalendarCycle Enum("CY", "MY", "CM", "CW", "WM", "WY", "DM", "CD",
561 "DY", "DW", "HD", "CH", "NH", "CN", "SN", "CS")
562
563 class PIVL_TS extends QSET_TS
564     alignment: CalendarCycle
565     count: INT

```

```

566         frequency: RTO
567         isFlexible: Boolean
568         period: PQ
569         phase: IVL_TS
570
571     class PQ extends QTY
572         codingRationale: CodingRationale
573         translation: Sequence(PQR)
574         unit: Code
575         value: Decimal
576         --operations
577         canonical : PQ
578         "+"(other : PQ) : PQ
579         "-"(other : PQ) : PQ
580         "-"() : PQ
581         "*" (other : PQ) : PQ
582         "*" (other : Real) : PQ
583         "*" (other : Real) : PQ
584         "*" (other : Integer) : PQ
585         "/"(other : PQ) : PQ
586         "/"(other : Real) : PQ
587         "/"(other : Real) : PQ
588         "/"(other : Integer) : PQ
589         inverted() : PQ
590         abs() : PQ
591         "min"(other : PQ) : PQ
592         "max"(other : PQ) : PQ
593         toInterval() : IVL_PQ
594         convert(unit: Code):PQ
595
596     class PQR extends CD
597         value: Decimal
598
599
600
601     class QSC_TS extends QSET_TS
602         code: CD
603
604     class QSD_PQ extends QSET_PQ
605         minuend: QSET_PQ
606         subtrahend: QSET_PQ
607
608     class QSD_TS extends QSET_TS
609         minuend: QSET_TS
610         subtrahend: QSET_TS
611
612     class QSET_CO extends ANY
613         originalText: ED
614
615     class QSET_INT extends ANY
616         originalText: ED
617
618     class QSET_MO extends ANY
619         originalText: ED
620
621     class QSET_PQ extends ANY
622         originalText: ED
623
624     class QSET_QTY extends ANY
625         originalText: ED
626
627     class QSET_REAL extends ANY
628         originalText: ED
629
630     class QSET_TS extends ANY
631         originalText: ED

```

```

632
633 class QSI_PQ extends QSET_PQ
634     term: Sequence(QSET_PQ)
635
636 class QSI_TS extends QSET_TS
637     term: Sequence(QSET_TS)
638
639 class QSP_PQ extends QSET_PQ
640     high: QSET_PQ
641     low: QSET_PQ
642
643 class QSP_TS extends QSET_TS
644     high: QSET_TS
645     low: QSET_TS
646
647 class QSS_PQ extends QSET_PQ
648     term: Sequence(PQ)
649
650 class QSS_TS extends QSET_TS
651     term: Sequence(TS)
652
653 class QSU_PQ extends QSET_PQ
654     term: Sequence(QSET_PQ)
655
656 class QSU_TS extends QSET_TS
657     term: Sequence(QSET_TS)
658
659 class UncertaintyType Enum("U", "N", "LN", "G", "E", "X2", "T", "F", "B")
660
661 class QTY extends ANY
662     expression: ED
663     originalText: ED
664     uncertainRange: IVL_QTY
665     uncertainty: QTY
666     uncertaintyType: UncertaintyType
667     --operations
668     "<"(other : QTY) : BL
669     "<="(other : QTY) : BL
670     ">="(other : QTY) : BL
671     ">"(other : QTY) : BL
672     "-"(other : QTY) : QTY
673     "+"(other : QTY) : QTY
674     comparable(other : QTY) : Boolean
675     isDifference(other : QTY) : Boolean
676
677 class REAL extends QTY
678     value: Decimal
679     --operations
680     "+"(other : REAL) : REAL
681     "-"(other : REAL) : REAL
682     "*" (other : REAL) : REAL
683     "/"(other : REAL) : REAL
684     "min"(other : REAL) : REAL
685     "max"(other : REAL) : REAL
686     "-"() : REAL
687     abs() : REAL
688     ceiling() : INT
689     floor() : INT
690     round() : INT
691     inverted() : REAL
692     power(other : REAL) : REAL
693     toInterval() : IVL_REAL
694
695 class RTO extends QTY
696     denominator: QTY
697     numerator: QTY

```

```

698
699     class SC extends ST
700         code: CD
701
702     class SD_TEXT extends ANY
703         ID: String
704         language: Code
705         mediaType: String
706         styleCode: set_Code
707
708     class SD_TITLE extends ANY
709         ID: String
710         language: Code
711         mediaType: String
712         styleCode: set_Code
713
714     class SLIST_INT extends ANY
715         digit: Sequence(INT)
716         origin: INT
717         scale: QTY
718
719     class SLIST_PQ extends ANY
720         digit: Sequence(INT)
721         origin: PQ
722         scale: QTY
723
724     class SLIST_REAL extends ANY
725         digit: Sequence(INT)
726         origin: REAL
727         scale: QTY
728
729     class SLIST_TS extends ANY
730         digit: Sequence(INT)
731         origin: TS
732         scale: QTY
733
734     class ST extends ANY
735         language: Code
736         translation: Sequence(ST)
737         value: String
738         --operations
739         mediaType() : String
740         size() : Integer
741         concat(other : ST) : ST
742         substring(lower : INT, upper : INT) : ST
743         toInteger() : INT
744         toReal() : REAL
745         toTimestamp() : TS
746
747     class TelecommunicationCapability Enum("voice", "fax", "data", "tty", "sms")
748
749     class TelecommunicationAddressUse Enum("H", "HP", "HV", "WP", "DIR", "PUB",
750 "BAD", "TMP", "AS", "EC", "MC", "PG")
751
752     class TEL extends ANY
753         capabilities: Set(TelecommunicationCapability)
754         use: Set(TelecommunicationAddressUse)
755         useablePeriod: QSET_TS
756         value: String
757         --operations
758         canonical() : TEL
759
760     class TS extends QTY
761         value: String
762         --operations
763         "+"(other : PQ) : TS

```

```
764         "-"(other : PQ) : TS
765         "-"(other : TS) : PQ
766         "min"(other : TS) : TS
767         "max"(other : TS) : TS
768         toInterval() : IVL_TS
769         precision() : Integer
770
771
772     class UVP_AD extends ANY
773         probability: Decimal
774         value: AD
775
776     class UVP_BL extends ANY
777         probability: Decimal
778         value: BL
779
780     class UVP_CD extends ANY
781         probability: Decimal
782         value: CD
783
784     class UVP_CO extends ANY
785         probability: Decimal
786         value: CO
787
788     class UVP_CS extends ANY
789         probability: Decimal
790         value: CS
791
792     class UVP_ED extends ANY
793         probability: Decimal
794         value: ED
795
796     class UVP_EN extends ANY
797         probability: Decimal
798         value: EN
799
800     class UVP_II extends ANY
801         probability: Decimal
802         value: II
803
804     class UVP_INT extends ANY
805         probability: Decimal
806         value: INT
807
808     class UVP_MO extends ANY
809         probability: Decimal
810         value: MO
811
812     class UVP_PQ extends ANY
813         probability: Decimal
814         value: PQ
815
816     class UVP_REAL extends ANY
817         probability: Decimal
818         value: REAL
819
820     class UVP_RTO extends ANY
821         probability: Decimal
822         value: RTO
823
824     class UVP_SC extends ANY
825         probability: Decimal
826         value: SC
827
828     class UVP_ST extends ANY
829         probability: Decimal
```

```
830         value: ST
831
832     class UVP_TEL extends ANY
833         probability: Decimal
834         value: TEL
835
836     class UVP_TS extends ANY
837         probability: Decimal
838         value: TS
839
840     class Uid extends String
841
842     class Uri extends String
843
844     class XReference
845         xref: String
846
847     class XmlID extends String
848
849     class XmlIDREF extends String
850
851     class set_Code
852         value: Set(Code)
853
854     class set_CodingRationale
855         value: Set(CodingRationale)
856
857     class set_EntityNamePartQualifier
858         value: Set(EntityNamePartQualifier)
859
860     class set_EntityNameUse
861         value: Set(EntityNameUse)
862
863     class set_IDREF
864         value: Set(XmlIDREF)
865
866     class set_TelecommunicationAddressUse
867         value: Set(TelecommunicationAddressUse)
868
869     class set_TelecommunicationCapability
870         value: Set(TelecommunicationCapability)
871
872     class COLL_AD extends ANY
873
874     class COLL_BL extends ANY
875
876     class COLL_CD extends ANY
877
878     class COLL_CO extends ANY
879
880     class COLL_CS extends ANY
881
882     class COLL_ED extends ANY
883
884     class COLL_EN extends ANY
885
886     class COLL_II extends ANY
887
888     class COLL_INT extends ANY
889
890     class COLL_MO extends ANY
891
892     class COLL_PQ extends ANY
893
894     class COLL_REAL extends ANY
895
```

```
896      class COLL_RTO extends ANY
897
898      class COLL_SC extends ANY
899
900      class COLL_ST extends ANY
901
902      class COLL_TEL extends ANY
903
904      class COLL_TS extends ANY
905
906      class BAG_AD extends COLL_AD
907        value: Bag(AD)
908
909      class BAG_BL extends COLL_BL
910        value: Bag(BL)
911
912      class BAG_CD extends COLL_CD
913        value: Bag(CD)
914
915      class BAG_CO extends COLL_CO
916        value: Bag(CO)
917
918      class BAG_CS extends COLL_CS
919        value: Bag(CS)
920
921      class BAG_ED extends COLL_ED
922        value: Bag(ED)
923
924      class BAG_EN extends COLL_EN
925        value: Bag(EN)
926
927      class BAG_II extends COLL_II
928        value: Bag(II)
929
930      class BAG_INT extends COLL_INT
931        value: Bag(INT)
932
933      class BAG_MO extends COLL_MO
934        value: Bag(MO)
935
936      class BAG_PQ extends COLL_PQ
937        value: Bag(PQ)
938
939      class BAG_REAL extends COLL_REAL
940        value: Bag(REAL)
941
942      class BAG_RTO extends COLL_RTO
943        value: Bag(RTO)
944
945      class BAG_SC extends COLL_SC
946        value: Bag(SC)
947
948      class BAG_ST extends COLL_ST
949        value: Bag(ST)
950
951      class BAG_TEL extends COLL_TEL
952        value: Bag(TEL)
953
954      class BAG_TS extends COLL_TS
955        value: Bag(TS)
956
957      class DSET_AD extends COLL_AD
958        value: Sequence(AD)
959
960      class DSET_BL extends COLL_BL
961        value: Sequence(BL)
```

```
962
963 class DSET_CD extends COLL_CD
964     value: Sequence(CD)
965
966 class DSET_CO extends COLL_CO
967     value: Sequence(CO)
968
969 class DSET_CS extends COLL_CS
970     value: Sequence(CS)
971
972 class DSET_ED extends COLL_ED
973     value: Sequence(ED)
974
975 class DSET_EN extends COLL_EN
976     value: Sequence(EN)
977
978 class DSET_II extends COLL_II
979     value: Sequence(II)
980
981 class DSET_INT extends COLL_INT
982     value: Sequence(INT)
983
984 class DSET_MO extends COLL_MO
985     value: Sequence(MO)
986
987 class DSET_PQ extends COLL_PQ
988     value: Sequence(PQ)
989
990 class DSET_REAL extends COLL_REAL
991     value: Sequence(REAL)
992
993 class DSET_RTO extends COLL_RTO
994     value: Sequence(RTO)
995
996 class DSET_SC extends COLL_SC
997     value: Sequence(SC)
998
999 class DSET_ST extends COLL_ST
1000     value: Sequence(ST)
1001
1002 class DSET_TEL extends COLL_TEL
1003     value: Sequence(TEL)
1004
1005 class DSET_TS extends COLL_TS
1006     value: Sequence(TS)
1007
1008 class HIST_AD extends LIST_AD
1009
1010 class HIST_BL extends LIST_BL
1011
1012 class HIST_CD extends LIST_CD
1013
1014 class HIST_CO extends LIST_CO
1015
1016 class HIST_CS extends LIST_CS
1017
1018 class HIST_ED extends LIST_ED
1019
1020 class HIST_EN extends LIST_EN
1021
1022 class HIST_II extends LIST_II
1023
1024 class HIST_INT extends LIST_INT
1025
1026 class HIST_MO extends LIST_MO
1027
```

```
1028     class HIST_PQ extends LIST_PQ
1029
1030     class HIST_REAL extends LIST_REAL
1031
1032     class HIST_RTO extends LIST_RTO
1033
1034     class HIST_SC extends LIST_SC
1035
1036     class HIST_ST extends LIST_ST
1037
1038     class HIST_TEL extends LIST_TEL
1039
1040     class HIST_TS extends LIST_TS
1041
1042     class LIST_AD extends COLL_AD
1043         value: Sequence(AD)
1044
1045     class LIST_BL extends COLL_BL
1046         value: Sequence(BL)
1047
1048     class LIST_CD extends COLL_CD
1049         value: Sequence(CD)
1050
1051     class LIST_CO extends COLL_CO
1052         value: Sequence(CO)
1053
1054     class LIST_CS extends COLL_CS
1055         value: Sequence(CS)
1056
1057     class LIST_ED extends COLL_ED
1058         value: Sequence(ED)
1059
1060     class LIST_EN extends COLL_EN
1061         value: Sequence(EN)
1062
1063     class LIST_II extends COLL_II
1064         value: Sequence(II)
1065
1066     class LIST_INT extends COLL_INT
1067         value: Sequence(INT)
1068
1069     class LIST_MO extends COLL_MO
1070         value: Sequence(MO)
1071
1072     class LIST_PQ extends COLL_PQ
1073         value: Sequence(PQ)
1074
1075     class LIST_REAL extends COLL_REAL
1076         value: Sequence(REAL)
1077
1078     class LIST_RTO extends COLL_RTO
1079         value: Sequence(RTO)
1080
1081     class LIST_SC extends COLL_SC
1082         value: Sequence(SC)
1083
1084     class LIST_ST extends COLL_ST
1085         value: Sequence(ST)
1086
1087     class LIST_TEL extends COLL_TEL
1088         value: Sequence(TEL)
1089
1090     class LIST_TS extends COLL_TS
1091         value: Sequence(TS)
1092
1093     class NPPD_AD extends ANY
```

```
1094         value: Sequence(UVP_AD)
1095
1096     class NPPD_BL extends ANY
1097         value: Sequence(UVP_BL)
1098
1099     class NPPD_CD extends ANY
1100         value: Sequence(UVP_CD)
1101
1102     class NPPD_CO extends ANY
1103         value: Sequence(UVP_CO)
1104
1105     class NPPD_CS extends ANY
1106         value: Sequence(UVP_CS)
1107
1108     class NPPD_ED extends ANY
1109         value: Sequence(UVP_ED)
1110
1111     class NPPD_EN extends ANY
1112         value: Sequence(UVP_EN)
1113
1114     class NPPD_II extends ANY
1115         value: Sequence(UVP_II)
1116
1117     class NPPD_INT extends ANY
1118         value: Sequence(UVP_INT)
1119
1120     class NPPD_MO extends ANY
1121         value: Sequence(UVP_MO)
1122
1123     class NPPD_PQ extends ANY
1124         value: Sequence(UVP_PQ)
1125
1126     class NPPD_REAL extends ANY
1127         value: Sequence(UVP_REAL)
1128
1129     class NPPD_RTO extends ANY
1130         value: Sequence(UVP_RTO)
1131
1132     class NPPD_SC extends ANY
1133         value: Sequence(UVP_SC)
1134
1135     class NPPD_ST extends ANY
1136         value: Sequence(UVP_ST)
1137
1138     class NPPD_TEL extends ANY
1139         value: Sequence(UVP_TEL)
1140
1141     class NPPD_TS extends ANY
1142         value: Sequence(UVP_TS)
1143
1144     class StrucDoc_Align Enum("left", "center", "right", "justify", "char")
1145
1146     class StrucDoc_Br
1147
1148     class StrucDoc_CMTtitle
1149         br: StrucDoc_Br
1150         content: StrucDoc_CMTtitle
1151         footnote: StrucDoc_TitleFootnote
1152         footnoteRef: StrucDoc_FootnoteRef
1153         linkHtml: StrucDoc_LinkHtml
1154         sub: StrucDoc_Sub
1155         sup: StrucDoc_Sup
1156
1157     class StrucDoc_Caption
1158
1159     class StrucDoc_Captioned
```

```

1160         caption: StrucDoc_Caption
1161
1162     class StrucDoc_CellScope Enum("row", "col", "rowgroup", "colgroup")
1163
1164     class StrucDoc_Col extends StrucDoc_ColItem
1165
1166     class StrucDoc_ColGroup extends StrucDoc_ColItem
1167         col: Sequence(StrucDoc_Col)
1168
1169     class StrucDoc_ColItem extends StrucDoc_TableItem
1170         span: Integer
1171         width: StrucDoc_Length
1172
1173     class StrucDoc_Content
1174         revised: StrucDoc_Revised
1175
1176     class StrucDoc_Footnote
1177
1178     class StrucDoc_FootnoteRef
1179         IDREF: String
1180
1181     class StrucDoc_Frame Enum("void", "above", "below", "hsides", "lhs", "rhs",
1182 "vsides", "box", "border")
1183
1184     class StrucDoc_Item extends StrucDoc_Captioned
1185
1186     class StrucDoc_Length extends String
1187
1188     class StrucDoc_LinkHtml
1189         href: String
1190         name: String
1191         rel: String
1192         rev: String
1193         title: String
1194
1195     class StrucDoc_List extends StrucDoc_Captioned
1196         value: Sequence(StrucDoc_Item)
1197         listType: StrucDoc_ListType
1198
1199     class StrucDoc_ListType Enum("ordered", "unordered")
1200
1201     class StrucDoc_Paragraph extends StrucDoc_Captioned
1202
1203     class StrucDoc_RenderMultiMedia
1204         caption: StrucDoc_Caption
1205         referencedObject: set_IDREF
1206
1207     class StrucDoc_Revised Enum("insert", "delete")
1208
1209     class StrucDoc_Rules Enum("none", "groups", "rows", "cols", "all")
1210
1211     class StrucDoc_Sub
1212
1213     class StrucDoc_Sup
1214
1215     class StrucDoc_TCell extends StrucDoc_TableItem
1216         abbr: String
1217         axis: String
1218         colspan: Integer
1219         headers: set_IDREF
1220         rowspan: Integer
1221         scope: StrucDoc_CellScope
1222
1223     class StrucDoc_TRow extends StrucDoc_TableItem
1224
1225     class StrucDoc_TRowGroup extends StrucDoc_TableItem

```

```

1226         tr: Sequence(StrucDoc_TRow)
1227
1228     class StrucDoc_Table extends StrucDoc_Captioned
1229         border: StrucDoc_Length
1230         cellpadding: StrucDoc_Length
1231         cellspacing: StrucDoc_Length
1232         col: Sequence(StrucDoc_Col)
1233         colgroup: Sequence(StrucDoc_ColGroup)
1234         frame: StrucDoc_Frame
1235         rules: StrucDoc_Rules
1236         summary: String
1237         tbody: Sequence(StrucDoc_TRowGroup)
1238         tfoot: StrucDoc_TRowGroup
1239         thead: StrucDoc_TRowGroup
1240         width: StrucDoc_Length
1241
1242     class StrucDoc_TableItem
1243         align: StrucDoc_Align
1244         char: String
1245         charoff: StrucDoc_Length
1246         valign: StrucDoc_VAlign
1247
1248     class StrucDoc_TitleFootnote
1249
1250     class StrucDoc_VAlign Enum("top", "middle", "bottom", "baseline")
1251
1252 class Factory
1253     AD(isNotOrdered: Boolean, part: Sequence(ADXP), use: Set(PostalAddressUse),
1254     useablePeriod: QSET_TS): AD
1255     ADXP(type: AddressPartType): ADXP
1256     ANY(flavorId: String, nullFlavor: NullFlavor, updateMode: UpdateMode): ANY
1257     BAG_AD(value: Bag(AD)): BAG_AD
1258     BAG_BL(value: Bag(BL)): BAG_BL
1259     BAG_CD(value: Bag(CD)): BAG_CD
1260     BAG_CO(value: Bag(CO)): BAG_CO
1261     BAG_CS(value: Bag(CS)): BAG_CS
1262     BAG_ED(value: Bag(ED)): BAG_ED
1263     BAG_EN(value: Bag(EN)): BAG_EN
1264     BAG_II(value: Bag(II)): BAG_II
1265     BAG_INT(value: Bag(INT)): BAG_INT
1266     BAG_MO(value: Bag(MO)): BAG_MO
1267     BAG_PQ(value: Bag(PQ)): BAG_PQ
1268     BAG_REAL(value: Bag(REAL)): BAG_REAL
1269     BAG_RTO(value: Bag(RTO)): BAG_RTO
1270     BAG_SC(value: Bag(SC)): BAG_SC
1271     BAG_ST(value: Bag(ST)): BAG_ST
1272     BAG_TEL(value: Bag(TEL)): BAG_TEL
1273     BAG_TS(value: Bag(TS)): BAG_TS
1274     BL(value: Boolean): BL
1275     CD(code: String, codeSystem: Uid, codeSystemName: String, codeSystemVersion:
1276     String, codingRationale: CodingRationale, displayName: ST, id: String,
1277     originalText: ED, source: XReference, translation: Sequence(CD), valueSet:
1278     Uid, valueSetVersion: String): CD
1279     CO(code: CD, value: Decimal): CO
1280     Code(ancestor: String): Code
1281     COLL_AD(ancestor: ANY): COLL_AD
1282     COLL_BL(ancestor: ANY): COLL_BL
1283     COLL_CD(ancestor: ANY): COLL_CD
1284     COLL_CO(ancestor: ANY): COLL_CO
1285     COLL_CS(ancestor: ANY): COLL_CS
1286     COLL_ED(ancestor: ANY): COLL_ED
1287     COLL_EN(ancestor: ANY): COLL_EN
1288     COLL_II(ancestor: ANY): COLL_II
1289     COLL_INT(ancestor: ANY): COLL_INT
1290     COLL_MO(ancestor: ANY): COLL_MO
1291     COLL_PQ(ancestor: ANY): COLL_PQ

```

1292 COLL\_REAL(ancestor: ANY): COLL\_REAL  
1293 COLL\_RTO(ancestor: ANY): COLL\_RTO  
1294 COLL\_SC(ancestor: ANY): COLL\_SC  
1295 COLL\_ST(ancestor: ANY): COLL\_ST  
1296 COLL\_TEL(ancestor: ANY): COLL\_TEL  
1297 COLL\_TS(ancestor: ANY): COLL\_TS  
1298 CS(code: String, codeSystem: Uid, codeSystemName: String, codeSystemVersion:  
1299 String): CS  
1300 Decimal(ancestor: Real): Decimal  
1301 DSET\_AD(value: Sequence(AD)): DSET\_AD  
1302 DSET\_BL(value: Sequence(BL)): DSET\_BL  
1303 DSET\_CD(value: Sequence(CD)): DSET\_CD  
1304 DSET\_CO(value: Sequence(CO)): DSET\_CO  
1305 DSET\_CS(value: Sequence(CS)): DSET\_CS  
1306 DSET\_ED(value: Sequence(ED)): DSET\_ED  
1307 DSET\_EN(value: Sequence(EN)): DSET\_EN  
1308 DSET\_II(value: Sequence(II)): DSET\_II  
1309 DSET\_INT(value: Sequence(INT)): DSET\_INT  
1310 DSET\_MO(value: Sequence(MO)): DSET\_MO  
1311 DSET\_PQ(value: Sequence(PQ)): DSET\_PQ  
1312 DSET\_REAL(value: Sequence(REAL)): DSET\_REAL  
1313 DSET\_RTO(value: Sequence(RTO)): DSET\_RTO  
1314 DSET\_SC(value: Sequence(SC)): DSET\_SC  
1315 DSET\_ST(value: Sequence(ST)): DSET\_ST  
1316 DSET\_TEL(value: Sequence(TEL)): DSET\_TEL  
1317 DSET\_TS(value: Sequence(TS)): DSET\_TS  
1318 ED(charset: Code, compression: Compression, data: String, description: ST,  
1319 integrityCheck: String, integrityCheckAlgorithm: IntegrityCheckAlgorithm,  
1320 language: Code, mediaType: String, reference: TEL, thumbnail: ED, translation:  
1321 Sequence(ED), value: String, xml: Any): ED  
1322 EIVL\_TS(event: TimingEvent, offset: IVL\_PQ): EIVL\_TS  
1323 EN(part: Sequence(ENXP), use: set\_EntityNameUse): EN  
1324 ENXP(qualifier: Set(EntityNamePartQualifier), type: EntityNamePartType):  
1325 ENXP  
1326 Factory(ancestor: Any): Factory  
1327 GLIST\_PQ(denominator: Integer, head: PQ, increment: QTY, period: Integer):  
1328 GLIST\_PQ  
1329 GLIST\_REAL(denominator: Integer, head: REAL, increment: QTY, period:  
1330 Integer): GLIST\_REAL  
1331 GLIST\_TS(denominator: Integer, head: TS, increment: QTY, period: Integer):  
1332 GLIST\_TS  
1333 HIST\_AD(ancestor: LIST\_AD): HIST\_AD  
1334 HIST\_BL(ancestor: LIST\_BL): HIST\_BL  
1335 HIST\_CD(ancestor: LIST\_CD): HIST\_CD  
1336 HIST\_CO(ancestor: LIST\_CO): HIST\_CO  
1337 HIST\_CS(ancestor: LIST\_CS): HIST\_CS  
1338 HIST\_ED(ancestor: LIST\_ED): HIST\_ED  
1339 HIST\_EN(ancestor: LIST\_EN): HIST\_EN  
1340 HIST\_II(ancestor: LIST\_II): HIST\_II  
1341 HIST\_INT(ancestor: LIST\_INT): HIST\_INT  
1342 HIST\_MO(ancestor: LIST\_MO): HIST\_MO  
1343 HIST\_PQ(ancestor: LIST\_PQ): HIST\_PQ  
1344 HIST\_REAL(ancestor: LIST\_REAL): HIST\_REAL  
1345 HIST\_RTO(ancestor: LIST\_RTO): HIST\_RTO  
1346 HIST\_SC(ancestor: LIST\_SC): HIST\_SC  
1347 HIST\_ST(ancestor: LIST\_ST): HIST\_ST  
1348 HIST\_TEL(ancestor: LIST\_TEL): HIST\_TEL  
1349 HIST\_TS(ancestor: LIST\_TS): HIST\_TS  
1350 HXIT(controlInformationExtension: String, controlInformationRoot: Uid,  
1351 validTimeHigh: String, validTimeLow: String): HXIT  
1352 II(displayable: Boolean, extension: String, identifierName: String,  
1353 reliability: IdentifierReliability, root: Uid, scope: IdentifierScope): II  
1354 INT(value: Integer): INT  
1355 IVL\_CO(any: CO, high: CO, highClosed: Boolean, low: CO, lowClosed: Boolean,  
1356 width: QTY): IVL\_CO  
1357 IVL\_INT(any: INT, high: INT, highClosed: Boolean, low: INT, lowClosed:

1358 Boolean, width: QTY): IVL\_INT  
1359 IVL\_MO(any: MO, high: MO, highClosed: Boolean, low: MO, lowClosed: Boolean,  
1360 width: QTY): IVL\_MO  
1361 IVL\_PQ(any: PQ, high: PQ, highClosed: Boolean, low: PQ, lowClosed: Boolean,  
1362 width: QTY): IVL\_PQ  
1363 IVL\_QTY(any: QTY, high: QTY, highClosed: Boolean, low: QTY, lowClosed:  
1364 Boolean, width: QTY): IVL\_QTY  
1365 IVL\_REAL(any: REAL, high: REAL, highClosed: Boolean, low: REAL, lowClosed:  
1366 Boolean, width: QTY): IVL\_REAL  
1367 IVL\_TS(any: TS, high: TS, highClosed: Boolean, low: TS, lowClosed: Boolean,  
1368 width: QTY): IVL\_TS  
1369 LIST\_AD(value: Sequence(AD)): LIST\_AD  
1370 LIST\_BL(value: Sequence(BL)): LIST\_BL  
1371 LIST\_CD(value: Sequence(CD)): LIST\_CD  
1372 LIST\_CO(value: Sequence(CO)): LIST\_CO  
1373 LIST\_CS(value: Sequence(CS)): LIST\_CS  
1374 LIST\_ED(value: Sequence(ED)): LIST\_ED  
1375 LIST\_EN(value: Sequence(EN)): LIST\_EN  
1376 LIST\_II(value: Sequence(II)): LIST\_II  
1377 LIST\_INT(value: Sequence(INT)): LIST\_INT  
1378 LIST\_MO(value: Sequence(MO)): LIST\_MO  
1379 LIST\_PQ(value: Sequence(PQ)): LIST\_PQ  
1380 LIST\_REAL(value: Sequence(REAL)): LIST\_REAL  
1381 LIST\_RTO(value: Sequence(RTO)): LIST\_RTO  
1382 LIST\_SC(value: Sequence(SC)): LIST\_SC  
1383 LIST\_ST(value: Sequence(ST)): LIST\_ST  
1384 LIST\_TEL(value: Sequence(TEL)): LIST\_TEL  
1385 LIST\_TS(value: Sequence(TS)): LIST\_TS  
1386 MO(currency: Code, value: Decimal): MO  
1387 NPPD\_AD(value: Sequence(UVP\_AD)): NPPD\_AD  
1388 NPPD\_BL(value: Sequence(UVP\_BL)): NPPD\_BL  
1389 NPPD\_CD(value: Sequence(UVP\_CD)): NPPD\_CD  
1390 NPPD\_CO(value: Sequence(UVP\_CO)): NPPD\_CO  
1391 NPPD\_CS(value: Sequence(UVP\_CS)): NPPD\_CS  
1392 NPPD\_ED(value: Sequence(UVP\_ED)): NPPD\_ED  
1393 NPPD\_EN(value: Sequence(UVP\_EN)): NPPD\_EN  
1394 NPPD\_II(value: Sequence(UVP\_II)): NPPD\_II  
1395 NPPD\_INT(value: Sequence(UVP\_INT)): NPPD\_INT  
1396 NPPD\_MO(value: Sequence(UVP\_MO)): NPPD\_MO  
1397 NPPD\_PQ(value: Sequence(UVP\_PQ)): NPPD\_PQ  
1398 NPPD\_REAL(value: Sequence(UVP\_REAL)): NPPD\_REAL  
1399 NPPD\_RTO(value: Sequence(UVP\_RTO)): NPPD\_RTO  
1400 NPPD\_SC(value: Sequence(UVP\_SC)): NPPD\_SC  
1401 NPPD\_ST(value: Sequence(UVP\_ST)): NPPD\_ST  
1402 NPPD\_TEL(value: Sequence(UVP\_TEL)): NPPD\_TEL  
1403 NPPD\_TS(value: Sequence(UVP\_TS)): NPPD\_TS  
1404 PIVL\_TS(alignment: CalendarCycle, count: INT, frequency: RTO, isFlexible:  
1405 Boolean, period: PQ, phase: IVL\_TS): PIVL\_TS  
1406 PQ(codingRationale: CodingRationale, translation: Sequence(PQR), unit: Code,  
1407 value: Decimal, canonical: PQ): PQ  
1408 PQR(value: Decimal): PQR  
1409 QSC\_TS(code: CD): QSC\_TS  
1410 QSD\_PQ(minuend: QSET\_PQ, subtrahend: QSET\_PQ): QSD\_PQ  
1411 QSD\_TS(minuend: QSET\_TS, subtrahend: QSET\_TS): QSD\_TS  
1412 QSET\_CO(originalText: ED): QSET\_CO  
1413 QSET\_INT(originalText: ED): QSET\_INT  
1414 QSET\_MO(originalText: ED): QSET\_MO  
1415 QSET\_PQ(originalText: ED): QSET\_PQ  
1416 QSET\_QTY(originalText: ED): QSET\_QTY  
1417 QSET\_REAL(originalText: ED): QSET\_REAL  
1418 QSET\_TS(originalText: ED): QSET\_TS  
1419 QSI\_PQ(term: Sequence(QSET\_PQ)): QSI\_PQ  
1420 QSI\_TS(term: Sequence(QSET\_TS)): QSI\_TS  
1421 QSP\_PQ(high: QSET\_PQ, low: QSET\_PQ): QSP\_PQ  
1422 QSP\_TS(high: QSET\_TS, low: QSET\_TS): QSP\_TS  
1423 QSS\_PQ(term: Sequence(PQ)): QSS\_PQ

1424 QSS\_TS(term: Sequence(TS)): QSS\_TS  
1425 QSU\_PQ(term: Sequence(QSET\_PQ)): QSU\_PQ  
1426 QSU\_TS(term: Sequence(QSET\_TS)): QSU\_TS  
1427 QTY(expression: ED, originalText: ED, uncertainRange: IVL\_QTY, uncertainty:  
1428 QTY, uncertaintyType: UncertaintyType): QTY  
1429 REAL(value: Decimal): REAL  
1430 RTO(denominator: QTY, numerator: QTY): RTO  
1431 SC(code: CD): SC  
1432 SD\_TEXT(ID: String, language: Code, mediaType: String, styleCode):  
1433 SD\_TEXT  
1434 SD\_TITLE(ID: String, language: Code, mediaType: String, styleCode:  
1435 set\_Code): SD\_TITLE  
1436 set\_Code(value: Set(Code)): set\_Code  
1437 set\_CodingRationale(value: Set(CodingRationale)): set\_CodingRationale  
1438 set\_EntityNamePartQualifier(value: Set(EntityNamePartQualifier)):  
1439 set\_EntityNamePartQualifier  
1440 set\_EntityNameUse(value: Set(EntityNameUse)): set\_EntityNameUse  
1441 set\_IDREF(value: Set(XmlIDREF)): set\_IDREF  
1442 set\_TelecommunicationAddressUse(value: Set(TelecommunicationAddressUse)):  
1443 set\_TelecommunicationAddressUse  
1444 set\_TelecommunicationCapability(value: Set(TelecommunicationCapability)):  
1445 set\_TelecommunicationCapability  
1446 SLIST\_INT(digit: Sequence(INT), origin: INT, scale: QTY): SLIST\_INT  
1447 SLIST\_PQ(digit: Sequence(INT), origin: PQ, scale: QTY): SLIST\_PQ  
1448 SLIST\_REAL(digit: Sequence(INT), origin: REAL, scale: QTY): SLIST\_REAL  
1449 SLIST\_TS(digit: Sequence(INT), origin: TS, scale: QTY): SLIST\_TS  
1450 ST(language: Code, translation: Sequence(ST), value: String): ST  
1451 StrucDoc\_Br(ancestor: Any): StrucDoc\_Br  
1452 StrucDoc\_Caption(ancestor: Any): StrucDoc\_Caption  
1453 StrucDoc\_Captioned(caption: StrucDoc\_Caption): StrucDoc\_Captioned  
1454 StrucDoc\_CMTitle(br: StrucDoc\_Br, content: StrucDoc\_CMTitle, footnote:  
1455 StrucDoc\_TitleFootnote, footnoteRef: StrucDoc\_FootnoteRef, linkHtml:  
1456 StrucDoc\_LinkHtml, sub: StrucDoc\_Sub, sup: StrucDoc\_Sup): StrucDoc\_CMTitle  
1457 StrucDoc\_Col(ancestor: StrucDoc\_ColItem): StrucDoc\_Col  
1458 StrucDoc\_ColGroup(col: Sequence(StrucDoc\_Col)): StrucDoc\_ColGroup  
1459 StrucDoc\_ColItem(span: Integer, width: StrucDoc\_Length): StrucDoc\_ColItem  
1460 StrucDoc\_Content(revised: StrucDoc\_Revised): StrucDoc\_Content  
1461 StrucDoc\_Footnote(ancestor: Any): StrucDoc\_Footnote  
1462 StrucDoc\_FootnoteRef(IDREF: String): StrucDoc\_FootnoteRef  
1463 StrucDoc\_Item(ancestor: StrucDoc\_Captioned): StrucDoc\_Item  
1464 StrucDoc\_Length(ancestor: String): StrucDoc\_Length  
1465 StrucDoc\_LinkHtml(href: String, name: String, rel: String, rev: String,  
1466 title: String): StrucDoc\_LinkHtml  
1467 StrucDoc\_List(value: Sequence(StrucDoc\_Item), listType: StrucDoc\_ListType):  
1468 StrucDoc\_List  
1469 StrucDoc\_Paragraph(ancestor: StrucDoc\_Captioned): StrucDoc\_Paragraph  
1470 StrucDoc\_RenderMultiMedia(caption: StrucDoc\_Caption, referencedObject:  
1471 set\_IDREF): StrucDoc\_RenderMultiMedia  
1472 StrucDoc\_Sub(ancestor: Any): StrucDoc\_Sub  
1473 StrucDoc\_Sup(ancestor: Any): StrucDoc\_Sup  
1474 StrucDoc\_Table(border: StrucDoc\_Length, cellpadding: StrucDoc\_Length,  
1475 cellspacing: StrucDoc\_Length, col: Sequence(StrucDoc\_Col), colgroup:  
1476 Sequence(StrucDoc\_ColGroup), frame: StrucDoc\_Frame, rules: StrucDoc\_Rules,  
1477 summary: String, tbody: Sequence(StrucDoc\_TRowGroup), tfoot:  
1478 StrucDoc\_TRowGroup, thead: StrucDoc\_TRowGroup, width: StrucDoc\_Length):  
1479 StrucDoc\_Table  
1480 StrucDoc\_TableItem(align: StrucDoc\_Align, char: String, charoff:  
1481 StrucDoc\_Length, valign: StrucDoc\_VAlign): StrucDoc\_TableItem  
1482 StrucDoc\_TCell(abbr: String, axis: String, colspan: Integer, headers:  
1483 set\_IDREF, rowspan: Integer, scope: StrucDoc\_CellScope): StrucDoc\_TCell  
1484 StrucDoc\_TitleFootnote(ancestor: Any): StrucDoc\_TitleFootnote  
1485 StrucDoc\_TRow(ancestor: StrucDoc\_TableItem): StrucDoc\_TRow  
1486 StrucDoc\_TRowGroup(tr: Sequence(StrucDoc\_TRow)): StrucDoc\_TRowGroup  
1487 TEL(capabilities: Set(TelecommunicationCapability), use:  
1488 Set(TelecommunicationAddressUse), useablePeriod: QSET\_TS, value: String): TEL  
1489 TS(value: String): TS

```
1490     Uid(ancestor: String): Uid
1491     Uri(ancestor: String): Uri
1492     UVP_AD(probability: Decimal, value: AD): UVP_AD
1493     UVP_BL(probability: Decimal, value: BL): UVP_BL
1494     UVP_CD(probability: Decimal, value: CD): UVP_CD
1495     UVP_CO(probability: Decimal, value: CO): UVP_CO
1496     UVP_CS(probability: Decimal, value: CS): UVP_CS
1497     UVP_ED(probability: Decimal, value: ED): UVP_ED
1498     UVP_EN(probability: Decimal, value: EN): UVP_EN
1499     UVP_II(probability: Decimal, value: II): UVP_II
1500     UVP_INT(probability: Decimal, value: INT): UVP_INT
1501     UVP_MO(probability: Decimal, value: MO): UVP_MO
1502     UVP_PQ(probability: Decimal, value: PQ): UVP_PQ
1503     UVP_REAL(probability: Decimal, value: REAL): UVP_REAL
1504     UVP_RTO(probability: Decimal, value: RTO): UVP_RTO
1505     UVP_SC(probability: Decimal, value: SC): UVP_SC
1506     UVP_ST(probability: Decimal, value: ST): UVP_ST
1507     UVP_TEL(probability: Decimal, value: TEL): UVP_TEL
1508     UVP_TS(probability: Decimal, value: TS): UVP_TS
1509     XmlID(ancestor: String): XmlID
1510     XmlIDREF(ancestor: String): XmlIDREF
1511     XP(code: String, codeSystem: String, codeSystemVersion: String, language:
1512 Code, nullFlavor: NullFlavor, value: String): XP
1513     XReference(xref: String): XReference
1514
1515     EndPackage
1516
```