GELLO REST API - Generic Version

Standard GELLO REST Query

A GELLO query can be made to the server at the following address with a HTTP POST message.

```
http://hostname:port/rest/gellov2/generic
```

or

```
https://hostname:tlsport/rest/gellov2/generic
```

POST request format using JSON

```
{
"Debug": string,
"Packages": [
{ "Name": string,
"Code": string
}
.....],
"ModelData": { ..... },
"Requests": [
{
"RequestId": string,
"GelloCode": string
},
.....]
}
```

Debug (optional) is character string with a list of debug options that control debug information returned in the reply.

```
Currently supported are
'V' - list the values of GELLO Let variables in the GELLO expression
'I' - list the imported GELLO packages used by the GELLO expression
'M' - return the full package RTTI for all packages used by the GELLO expression
```

ModelData (optional) If the GELLO expression contains a Context statement, the data for the context can be supplied. It must have the same GELLO type as that of the Context statement. Errors may be reported, or data may not be visible if the class of the Context statement does not match that of the **Mode lData.**

Packages (optional) is an array of objects containing package name and source code for supplied package. This can be used to supply additional packages for loading model data and for generating GELLO results. Note - only class attributes (properties) will have any meaning in the uploaded packages, and also operations (methods) which are written in pure GELLO. Any operations which are defined by an embedded package will not be operate correctly, and most likely return null GELLO values.

Requests (required) is a JSON array containing a list of GELLO queries to be made against the Patient Context

Each request contains the following JSON object

RequestId (required) is a unique Identifier supplied by the user to identify multiple jobs within the request. It should not be empty or null. Each request should contain a unique ID.

GelloCode (required) is a string containing a valid GELLO query to be executed in the HL7 context. It may contain contain white space and line breaks, but these should be escaped according to JSON rules.

POST reply from server

```
{
"Results": [
{
"debugVarTypes": [string, .....],
"debugVars": { .... }
"debugImports" [string, .....]
"requestId": string,
"wasSuccessful": boolean,
"errors": [string, .....],
"result": { ..... },
},
.....]
}
```

Results (required) is a JSON array containing a list of results from the GELLO queries made.

Each result contains the following JSON object. Results may appear out of order but can be collated via the **RequestId** parameter.

requestId (required) matches one and only one requestId in the Request.

wasSuccessful (required) contains boolean true or false, representing whether the GELLO query was successful or not.

result (optional) contains the result of the GELLO query in JSON format. The contents of this is dependent on the nature of the GELLO query. It may be missing or null if the GELLO expression failed.

errors (optional) contains a string list of any errors encountered in compiling and/or executing the GELLO query, and should contain at least one item if **wasSuccessful** returned false. It may be missing or empty if the GELLO expression was successful.

The following parameters will be returned when Debugging options are set.

debugVarTypes (optional) a list of formal type names for the debugVars as seen by the compiler.

debugVars (optional) a tuple object containing a list of the Let variables.

debugImports (optional) a list of package names used by the GELLO expression.

Example

Here is some example Gello you can try:

```
Let x:Integer = 50
Let a:Integer = 10
Let y:Real = ((a + x)*2) min 10
Let yy:Real = -y max 20
Let s:String = "12312313"
Let z:Real = y + 20.5
Let s1:String = s.concat(' ').concat(z.toChar())
s1
```

Request

```
{
"Debug": "V,I",
"Packages": [
],
"Requests": [
{
"RequestId": "1",
"GelloCode": "Let x:Integer = 50\r\nLet a:Integer = 10\r\nLet y:Real = ((a
+ x)*2) min 10\r\nLet yy:Real = -y max 20\r\nLet s:String = \"12312313\"
\r\nLet z:Real = y + 20.5\r\nLet s1:String = s.concat(' ').concat(z.
toChar())\r\ns1\r\n"
}
]
}
```

Reply

```
{
"Results": [
{
"requestID": "1",
"debugVarTypes": [
"Integer",
"Integer",
"Integer",
"Real",
"String",
"Real",
"String",
"String"
],
"debugVars": {
"#type": "Tuple",
"x": 50,
"a": 10,
"y": 10,
"yy": 20,
"s": "12312313",
"z": 30.5,
"s1": "12312313 30.5",
"_Result_": "12312313 30.5"
},
"debugImports": [
"System",
"Main"
],
"wasSuccessful": true,
"result": "12312313 30.5"
}
]
}
```

Formatting

Requests and Responses are formatted using JSON. Fields of JSON objects may appear in a different order to that documented. However for efficient JSON streaming in the future, some fields should be in the preferred order. In particular, Packages should precede ModelData, and ModelData should precede GELLO requests.

Compact JSON type inference[edit]
Normally, the GELLO type for data will be known before loading a JSON value into a class, however there are occasions where OCL types can be inferred from the JSON output. The following rules describe how these OCL types can be inferred.

JSON **number** ---> OCL **"Real"**

JSON **string** ---> OCL **"String"**

JSON **boolean** (true/false) ---> OCL **"Boolean"**

JSON **null** ---> OCL **"Any"** with oclIsDefined property = False

JSON **array** ---> OCL **"Sequence(Any)".** Usually the element type can be inferred also however, the array may contain mixed types.

JSON **object** ---> OCL class with class name contained in the "#type" class property. If this is omitted or of value "Tuple" it is assumed to be an OCL **Tuple.**

There are some minor exceptions to the above. A **Real** may have the values NaN, +Inf and -Inf, in which case it is represented as a OCL class of type **"Real"** with a sub property of "value" containing the string representation of the exceptional Real value. An **Integer** is descendant of class **Real,** and could be inferred from the additional regex **"\-?[0-9]+/"**, however JSON does not make this distinction, and the GELLO engine will perform the necessary conversion if required.

A null class may be defined by creating an object with the elements #type = <class name> and #undefined = true. Optionally one can add exception messages with #exception = string

The primitive OCL data types String, Real and Boolean are represented directly as their respective JSON values with the Integer data type encoded as a JSON number, except when the formal and actual types differ, in which case a JSON object wrapper is required with an explicit "#type" metadata. Unrepresentable JSON primitive values (e.g. NaN, Inf, etc) are represented as a JSON object with key /string pair "value":"literal-value", possibly with a "#type" metadata pair. This also applies to the Gello Boolean "unknown" value. OCL Class values are represented as JSON objects with OCL attributes being represented as JSON key/value pairs with null attributes being omitted. When the type of an OCL class type differs from its implied type (e.g. the declared attribute type of its parent), the JSON object includes a "#type" : "ClassName" metadata pair for the object, otherwise the type of the class is implied by its context within the JSON structure. Exceptional OCL values (i.e. undefined) are represented as a JSON object with the metadata "#undefined":true, with optional "#exception":"exception message". OCL Collections (Set, Bag, Sequence) are represented as JSON arrays with the elements being JSON values. If the collection class differs from the implied contextual collection class, it is deduced from the first element of the collection. Any null collection elements must be encoded as a JSON null value.

Additional GELLO IDE support requests

When used in conjunction with the GELLO IDE, some additional requests can be made. The remote GELLO server and the GELLO IDE may have different packages automatically loaded. The installedmodels request can be used to determine which models the server is built with.

```
http://hostname:port/rest/gellov2/installedmodels
```

or

```
https://hostname:tlsport/rest/gellov2/installedmodels
```

request format

```
{
"namesOnly": boolean,
"asGello": boolean
}
```

namesOnly (required) list only the names of installed packages

asGello (required when **namesOnly** = false)

Reply format

```
{
"wasSuccessful": boolean,
"names": [ string, .... ]
"packages": [
{
"Name": string,
"Code": string
},
"models": { ..... }
.....]
```

wasSuccessful (returned always), returns whether query was successful.

names (returned when **namesOnly** = true), returns a list of installed packages.

packages (returned when **namesOnly** = false and **asGello** = true), a full listing of installed packages in Gello Package format.

```
Name - package name
Code - package source code
```

models (returned when **namesOnly** = false and **asGello** = false), a full listing of installed packages in GELLO RTTI format (currently undocumented and subject to change).

Example - list installed package names only

Request

```
{
"namesOnly": true
}
```

Reply

```
{
"wasSuccessful": true,
"names": [
"System",
"XMLAny",
"iso_21090_datatypes"
]
}
```

```
 {
 "wasSuccessful": true,
 "packages": [
  {
```

```
   "Name": "System",
   "Code": "Package System\r\n\r\n  class T extends Any\r\n\r\n  class
_Undefined_\r\n\r\n  class _Untyped_\r\n\r\n  class Any\r\n\r\n    --
methods\r\n    allInstances(): Set(T)\r\n    oclTypeName(): String\r\n
oclAsType(t: T): T\r\n    oclIsDefined(): Boolean\r\n    oclIsInState
(statename: String): Boolean\r\n    oclIsKindOf(baseType: T):
Boolean\r\n    oclType(): T\r\n    =(o: Any): Boolean\r\n    <>(o: Any):
Boolean\r\n\r\n  class Real extends Any\r\n\r\n    -- methods\r\n    abs():
Real\r\n    ceiling(): Integer\r\n    floor(): Integer\r\n    inverted():
Real\r\n    max(rhs: Real): Real\r\n    min(rhs: Real): Real\r\n    power
(rhs: Real): Real\r\n    round(): Integer\r\n    toChar(): String\r\n
toCharFormat(decimalPlaces: Integer): String\r\n    +(rhs: Real):
Real\r\n    /(rhs: Real): Real\r\n    =(rhs: Real): Boolean\r\n    >=(rhs:
Real): Boolean\r\n    >(rhs: Real): Boolean\r\n    <=(rhs: Real):
Boolean\r\n    <(rhs: Real): Boolean\r\n    *(rhs: Real): Real\r\n    -():
Real\r\n    <>(rhs: Real): Boolean\r\n    -(rhs: Real): Real\r\n\r\n
class String extends Any\r\n\r\n    -- methods\r\n    concat(rhs: String):
String\r\n    lpad(size: Integer, padChar: String): String\r\n    ltrim
(trimChars: String): String\r\n    replace(replaceThis: String, withThis:
String): String\r\n    rpad(size: Integer, padChar: String): String\r\n
rtrim(trimChars: String): String\r\n    size(): Integer\r\n    substring
(lower: Integer, upper: Integer): String\r\n    toInteger():
Integer\r\n    toLower(): String\r\n    toReal(): Real\r\n    toUpper():
String\r\n    =(rhs: String): Boolean\r\n    <>(rhs: String):
Boolean\r\n\r\n  class Boolean extends Any\r\n\r\n    -- methods\r\n
asOCLString(): String\r\n    implies(rhs: Boolean): Boolean\r\n    and
(rhs: Boolean): Boolean\r\n    =(rhs: Boolean): Boolean\r\n    <>(rhs:
Boolean): Boolean\r\n    not(): Boolean\r\n    or(rhs: Boolean):
Boolean\r\n    xor(rhs: Boolean): Boolean\r\n\r\n  class Integer extends
Real\r\n\r\n    -- methods\r\n    abs(): Integer\r\n    max(rhs: Integer):
Integer\r\n    min(rhs: Integer): Integer\r\n    toChar(): String\r\n    +
(rhs: Integer): Integer\r\n    div(rhs: Integer): Integer\r\n    mod(rhs:
Integer): Integer\r\n    *(rhs: Integer): Integer\r\n    -():
Integer\r\n    -(rhs: Integer): Integer\r\n\r\n  class Base64 extends
Any\r\n\r\n  class Collection(T) extends Any\r\n\r\n    -- methods\r\n
excludes(o: Any): Boolean\r\n    excludesAll(c: Collection(T)):
Boolean\r\n    includes(o: Any): Boolean\r\n    includesAll(c: Collection
(T)): Boolean\r\n    exists(e: Boolean): Boolean\r\n    forAll(e:
Boolean): Boolean\r\n    isEmpty(): Boolean\r\n    Iterator(stable:
Boolean): Iterator\r\n    notEmpty(): Boolean\r\n    size():
Integer\r\n    sort(): Sequence(T)\r\n    sum(): Any\r\n    sortedBy(e:
Any): Sequence(T)\r\n    count(o: Any): Integer\r\n    =(o: Collection
(T)): Boolean\r\n    <>(o: Collection(T)): Boolean\r\n\r\n  class Set(T)
extends Collection(T)\r\n\r\n    -- methods\r\n    select(e: Boolean): Set
(T)\r\n    reject(e: Boolean): Set(T)\r\n    including(e: Any): Set(T)
\r\n    excluding(e: Any): Set(T)\r\n    flatten(): Set(T)\r\n    union(b:
Bag(T)): Bag(T)\r\n    union(s: Set(T)): Set(T)\r\n    intersection(b: Bag
(T)): Set(T)\r\n    intersection(s: Set(T)): Set(T)\r\n
symmetricDifference(s: Set(T)): Set(T)\r\n    collect(e: Any): Bag(T)
\r\n    -(s: Set(T)): Set(T)\r\n\r\n  class Bag(T) extends Collection(T)
\r\n\r\n    -- methods\r\n    excluding(e: Any): Bag(T)\r\n    flatten():
Bag(T)\r\n    including(e: Any): Bag(T)\r\n    intersection(s: Set(T)): Set
(T)\r\n    intersection(b: Bag(T)): Bag(T)\r\n    union(s: Set(T)): Bag(T)
\r\n    union(b: Bag(T)): Bag(T)\r\n    reject(e: Boolean): Bag(T)\r\n
select(e: Boolean): Bag(T)\r\n    collect(e: Any): Bag(T)\r\n\r\n  class
Sequence(T) extends Collection(T)\r\n\r\n    -- methods\r\n    append(o:
Any): Sequence(T)\r\n    prepend(o: Any): Sequence(T)\r\n    at(i:
Integer): T\r\n    subSequence(lower: Integer, upper: Integer): Sequence(T)
\r\n    excluding(e: Any): Sequence(T)\r\n    first(): T\r\n    last():
T\r\n    flatten(): Sequence(T)\r\n    including(e: Any): Sequence(T)
\r\n    indexOf(o: Any): Integer\r\n    insertAt(i: Integer, o: Any):
Sequence(T)\r\n    reverse(): Sequence(T)\r\n    union(b: Sequence(T)):
Sequence(T)\r\n    collect(e: Any): Sequence(T)\r\n    reject(e: Boolean):
Sequence(T)\r\n    select(e: Boolean): Sequence(T)\r\n\r\n  Tuple{\r\n  }
\r\n\r\n  class Iterator extends Any\r\n\r\n    -- methods\r\n
get_element(): Any\r\n    is_stable(): Boolean\r\n\r\n\r\nEndPackage\r\n"
  },
  {
   "Name": "XMLAny",
   "Code": "Package XMLAny\r\n\r\n  class xmlAttribute extends Any\r\n
name: String\r\n    value: String\r\n\r\n  class xmlNode extends
```

```
Any\r\n    name: String\r\n    text_parts: Sequence(String)\r\n
attributes: Sequence(xmlAttribute)\r\n    nodes: Sequence(xmlNode)
\r\n\r\n    -- methods\r\n    text(): String\r\n\r\n\r\nEndPackage\r\n"
  },
  {
   "Name": "iso_21090_datatypes",
   "Code": "Package iso_21090_datatypes\r\n\r\n  class NullFlavor Enum(\"
NI\", \"INV\", \"OTH\", \"NINF\", \"PINF\", \"UNC\", \"DER\", \"UNK\", \"
ASKU\", \"NAV\", \"QS\", \"NASK\", \"TRC\", \"MSK\", \"NA\")\r\n\r\n
class XP extends Any\r\n    code: String\r\n    codeSystem: String\r\n
codeSystemVersion: String\r\n    language: String\r\n    nullFlavor:
NullFlavor\r\n    value: String\r\n\r\n  class ANY extends Any\r\n
flavorId: String\r\n    nullFlavor: NullFlavor\r\n\r\n    --
methods\r\n    equals(other: ANY): BL\r\n    isNull(): BL\r\n    notNull():
BL\r\n\r\n  class BL extends ANY\r\n    value: Boolean\r\n\r\n    --
methods\r\n    asBoolean(): Boolean\r\n    and(other: BL): BL\r\n    or
(other: BL): BL\r\n    xor(other: BL): BL\r\n    implies(other: BL):
BL\r\n    not(): BL\r\n\r\n  class CD extends ANY\r\n    code:
String\r\n    codeSystem: String\r\n    codeSystemName: String\r\n
codeSystemVersion: String\r\n    codingRationale: String\r\n
displayName: ST\r\n    id: String\r\n    originalText: ED\r\n    source:
XReference\r\n    translation: Sequence(CD)\r\n    valueSet: String\r\n
valueSetVersion: String\r\n\r\n    -- methods\r\n    implies(other: CS):
BL\r\n    implies(other: CD): BL\r\n\r\n  class QTY extends ANY\r\n
expression: ED\r\n    originalText: ED\r\n    uncertainRange:
IVL_QTY\r\n    uncertainty: QTY\r\n    uncertaintyType: String\r\n\r\n
-- methods\r\n    <(other: QTY): Boolean\r\n    <=(other: QTY):
Boolean\r\n    >=(other: QTY): Boolean\r\n    >(other: QTY):
Boolean\r\n    -(other: QTY): QTY\r\n    +(other: QTY): QTY\r\n
comparable(other: QTY): BL\r\n    isDifference(other: QTY): BL\r\n\r\n
class COLL_T extends ANY\r\n\r\n    -- methods\r\n    size(): INT\r\n
includes(o: ANY): BL\r\n    excludes(o: ANY): BL\r\n    count(o: ANY):
INT\r\n    includesAll(other: Collection(T)): BL\r\n    excludesAll(other:
Collection(T)): BL\r\n    isEmpty(): BL\r\n    notEmpty(): BL\r\n\r\n
class BAG_T extends COLL_T\r\n    item: Bag(T)\r\n\r\n    --
methods\r\n    union(other: BAG_T): BAG_T\r\n    union(other: DSET_T):
BAG_T\r\n    intersection(other: BAG_T): BAG_T\r\n    intersection(other:
DSET_T): DSET_T\r\n    including(o: ANY): BAG_T\r\n    including(other:
COLL_T): BAG_T\r\n    excluding(o: ANY): BAG_T\r\n    excluding(other:
COLL_T): BAG_T\r\n    asSet(): DSET_T\r\n    asList(): LIST_T\r\n\r\n
class DSET_T extends COLL_T\r\n    item: Bag(T)\r\n\r\n    --
methods\r\n    union(other: DSET_T): DSET_T\r\n    union(other: BAG_T):
BAG_T\r\n    intersection(other: DSET_T): DSET_T\r\n    intersection
(other: BAG_T): DSET_T\r\n    -(other: DSET_T): DSET_T\r\n    including(o:
ANY): DSET_T\r\n    including(other: DSET_T): DSET_T\r\n    excluding(o:
ANY): DSET_T\r\n    excluding(other: DSET_T): DSET_T\r\n
symmetricDifference(other: DSET_T): DSET_T\r\n    asBag(): BAG_T\r\n
asList(): LIST_T\r\n\r\n  class LIST_T extends COLL_T\r\n    item: Sequence
(T)\r\n\r\n    -- methods\r\n    append(other: LIST_T): LIST_T\r\n
append(other: DSET_T): LIST_T\r\n    append(other: ANY): LIST_T\r\n
prepend(p1: ANY): LIST_T\r\n    insertAt(o: ANY, position: Integer):
LIST_T\r\n    subList(p1: Integer, p2: Integer): LIST_T\r\n    at(index:
Integer): ANY\r\n    indexOf(o: ANY): Integer\r\n    first(): ANY\r\n
last(): ANY\r\n    tail(): LIST_T\r\n    including(o: ANY): LIST_T\r\n
excluding(o: ANY): LIST_T\r\n    asBag(): BAG_T\r\n    asSet():
DSET_T\r\n\r\n  class BAG_AD extends BAG_T\r\n    item: Bag(AD)\r\n\r\n
class BAG_BL extends BAG_T\r\n    item: Bag(BL)\r\n\r\n  class BAG_CD
extends BAG_T\r\n    item: Bag(CD)\r\n\r\n  class BAG_CO extends
BAG_T\r\n    item: Bag(CO)\r\n\r\n  class BAG_CS extends BAG_T\r\n
item: Bag(CS)\r\n\r\n  class BAG_ED extends BAG_T\r\n    item: Bag(ED)
\r\n\r\n  class BAG_EN extends BAG_T\r\n    item: Bag(EN)\r\n\r\n  class
BAG_II extends BAG_T\r\n    item: Bag(II)\r\n\r\n  class BAG_INT extends
BAG_T\r\n    item: Bag(INT)\r\n\r\n  class BAG_MO extends BAG_T\r\n
item: Bag(MO)\r\n\r\n  class BAG_PQ extends BAG_T\r\n    item: Bag(PQ)
\r\n\r\n  class BAG_REAL extends BAG_T\r\n    item: Bag(REAL)\r\n\r\n
class BAG_RTO extends BAG_T\r\n    item: Bag(RTO)\r\n\r\n  class BAG_SC
extends BAG_T\r\n    item: Bag(SC)\r\n\r\n  class BAG_ST extends
BAG_T\r\n    item: Bag(ST)\r\n\r\n  class BAG_TEL extends BAG_T\r\n
item: Bag(TEL)\r\n\r\n  class BAG_TS extends BAG_T\r\n    item: Bag(TS)
\r\n\r\n  class CS extends ANY\r\n    code: String\r\n    codeSystem:
String\r\n    codeSystemName: String\r\n    codeSystemVersion:
```

```
String\r\n\r\n    -- methods\r\n    implies(other: CS): BL\r\n    implies
(other: CD): BL\r\n\r\n  class DSET_AD extends DSET_T\r\n    item: Sequence
(AD)\r\n\r\n  class DSET_BL extends DSET_T\r\n    item: Sequence(BL)
\r\n\r\n  class DSET_CD extends DSET_T\r\n    item: Sequence(CD)\r\n\r\n
class DSET_CO extends DSET_T\r\n    item: Sequence(CO)\r\n\r\n  class
DSET_CS extends DSET_T\r\n    item: Sequence(CS)\r\n\r\n  class DSET_ED
extends DSET_T\r\n    item: Sequence(ED)\r\n\r\n  class DSET_EN extends
DSET_T\r\n    item: Sequence(EN)\r\n\r\n  class DSET_II extends
DSET_T\r\n    item: Sequence(II)\r\n\r\n  class DSET_INT extends
DSET_T\r\n    item: Sequence(INT)\r\n\r\n  class DSET_MO extends
DSET_T\r\n    item: Sequence(MO)\r\n\r\n  class DSET_PQ extends
DSET_T\r\n    item: Sequence(PQ)\r\n\r\n  class DSET_REAL extends
DSET_T\r\n    item: Sequence(REAL)\r\n\r\n  class DSET_RTO extends
DSET_T\r\n    item: Sequence(RTO)\r\n\r\n  class DSET_SC extends
DSET_T\r\n    item: Sequence(SC)\r\n\r\n  class DSET_ST extends
DSET_T\r\n    item: Sequence(ST)\r\n\r\n  class DSET_TEL extends
DSET_T\r\n    item: Sequence(TEL)\r\n\r\n  class DSET_TS extends
DSET_T\r\n    item: Sequence(TS)\r\n\r\n  class ED extends ANY\r\n
charset: String\r\n    compression: String\r\n    data: String\r\n
description: ST\r\n    integrityCheck: String\r\n
integrityCheckAlgorithm: String\r\n    language: String\r\n    mediaType:
String\r\n    reference: TEL\r\n    thumbnail: ED\r\n    translation:
Sequence(ED)\r\n    value: String\r\n    xml: xmlNode\r\n\r\n  class
QSET_TS extends ANY\r\n    originalText: ED\r\n\r\n  class EN extends
ANY\r\n    part: Sequence(ENXP)\r\n    use: Set(EntityNameUse)\r\n\r\n
-- methods\r\n    canonical(): EN\r\n\r\n  class ENXP extends XP\r\n
qualifier: Set(EntityNamePartQualifier)\r\n    type: String\r\n\r\n  class
Factory extends Any\r\n\r\n    -- methods\r\n    Now(): TS\r\n    AD
(isNotOrdered: Boolean, part: Sequence(ADXP), use: Set(PostalAddressUse),
useablePeriod: QSET_TS): AD\r\n    ADXP(_type_: AddressPartType):
ADXP\r\n    ANY(flavorId: String, nullFlavor: NullFlavor, updateMode:
String): ANY\r\n    BAG_AD(value: Bag(AD)): BAG_AD\r\n    BAG_BL(value: Bag
(BL)): BAG_BL\r\n    BAG_CD(value: Bag(CD)): BAG_CD\r\n    BAG_CO(value:
Bag(CO)): BAG_CO\r\n    BAG_CS(value: Bag(CS)): BAG_CS\r\n    BAG_ED
(value: Bag(ED)): BAG_ED\r\n    BAG_EN(value: Bag(EN)): BAG_EN\r\n
BAG_II(value: Bag(II)): BAG_II\r\n    BAG_INT(value: Bag(INT)):
BAG_INT\r\n    BAG_MO(value: Bag(MO)): BAG_MO\r\n    BAG_PQ(value: Bag
(PQ)): BAG_PQ\r\n    BAG_REAL(value: Bag(REAL)): BAG_REAL\r\n    BAG_RTO
(value: Bag(RTO)): BAG_RTO\r\n    BAG_SC(value: Bag(SC)): BAG_SC\r\n
BAG_ST(value: Bag(ST)): BAG_ST\r\n    BAG_TEL(value: Bag(TEL)):
BAG_TEL\r\n    BAG_TS(value: Bag(TS)): BAG_TS\r\n    BL(value: Boolean):
BL\r\n    CD(code: String, codeSystem: String, codeSystemName: String,
codeSystemVersion: String, codingRationale: String, displayName: ST, id:
String, originalText: ED, source: XReference, translation: Sequence(CD),
valueSet: String, valueSetVersion: String): CD\r\n    CD_SNOMED_CT
(snomedCode: String): CD\r\n    CD_LOINC(loincCode: String): CD\r\n
CD_LOCAL(localCode: String, codeSystemName: String): CD\r\n    CO(code:
CD, value: Real): CO\r\n    Code(ancestor: String): String\r\n    CS(code:
String, codeSystem: String, codeSystemName: String, codeSystemVersion:
String): CS\r\n    Decimal(ancestor: Real): Real\r\n    DSET_AD(value:
Sequence(AD)): DSET_AD\r\n    DSET_BL(value: Sequence(BL)): DSET_BL\r\n
DSET_CD(value: Sequence(CD)): DSET_CD\r\n    DSET_CO(value: Sequence(CO)):
DSET_CO\r\n    DSET_CS(value: Sequence(CS)): DSET_CS\r\n    DSET_ED(value:
Sequence(ED)): DSET_ED\r\n    DSET_EN(value: Sequence(EN)): DSET_EN\r\n
DSET_II(value: Sequence(II)): DSET_II\r\n    DSET_INT(value: Sequence
(INT)): DSET_INT\r\n    DSET_MO(value: Sequence(MO)): DSET_MO\r\n
DSET_PQ(value: Sequence(PQ)): DSET_PQ\r\n    DSET_REAL(value: Sequence
(REAL)): DSET_REAL\r\n    DSET_RTO(value: Sequence(RTO)): DSET_RTO\r\n
DSET_SC(value: Sequence(SC)): DSET_SC\r\n    DSET_ST(value: Sequence(ST)):
DSET_ST\r\n    DSET_TEL(value: Sequence(TEL)): DSET_TEL\r\n    DSET_TS
(value: Sequence(TS)): DSET_TS\r\n    ED(charset: String, compression:
String, data: String, description: ST, integrityCheck: String,
integrityCheckAlgorithm: String, language: String, mediaType: String,
reference: TEL, thumbnail: ED, translation: Sequence(ED), value: String,
xml: xmlNode): ED\r\n    EIVL_TS(event: String, offset: IVL_PQ):
EIVL_TS\r\n    EN(part: Sequence(ENXP), use: Set(EntityNameUse)):
EN\r\n    ENXP(qualifier: Set(EntityNamePartQualifier), _type_: String):
ENXP\r\n    GLIST_PQ(denominator: Integer, head: PQ, increment: QTY,
period: Integer): GLIST_PQ\r\n    GLIST_REAL(denominator: Integer, head:
REAL, increment: QTY, period: Integer): GLIST_REAL\r\n    GLIST_TS
(denominator: Integer, head: TS, increment: QTY, period: Integer):
```

```
GLIST_TS\r\n    HIST_AD(ancestor: LIST_AD): LIST_AD\r\n    HIST_BL
(ancestor: LIST_BL): LIST_BL\r\n    HIST_CD(ancestor: LIST_CD):
LIST_CD\r\n    HIST_CO(ancestor: LIST_CO): LIST_CO\r\n    HIST_CS
(ancestor: LIST_CS): LIST_CS\r\n    HIST_ED(ancestor: LIST_ED):
LIST_ED\r\n    HIST_EN(ancestor: LIST_EN): LIST_EN\r\n    HIST_II
(ancestor: LIST_II): LIST_II\r\n    HIST_INT(ancestor: LIST_INT):
LIST_INT\r\n    HIST_MO(ancestor: LIST_MO): LIST_MO\r\n    HIST_PQ
(ancestor: LIST_PQ): LIST_PQ\r\n    HIST_REAL(ancestor: LIST_REAL):
LIST_REAL\r\n    HIST_RTO(ancestor: LIST_RTO): LIST_RTO\r\n    HIST_SC
(ancestor: LIST_SC): LIST_SC\r\n    HIST_ST(ancestor: LIST_ST):
LIST_ST\r\n    HIST_TEL(ancestor: LIST_TEL): LIST_TEL\r\n    HIST_TS
(ancestor: LIST_TS): LIST_TS\r\n    HXIT(controlInformationExtension:
String, controlInformationRoot: String, validTimeHigh: String,
validTimeLow: String): Any\r\n    II(displayable: Boolean, extension:
String, identifierName: String, reliability: String, root: String, scope:
String): II\r\n    INT(value: Integer): INT\r\n    IVL_CO(any: CO, high:
CO, highClosed: Boolean, low: CO, lowClosed: Boolean, width: QTY):
IVL_CO\r\n    IVL_INT(any: INT, high: INT, highClosed: Boolean, low: INT,
lowClosed: Boolean, width: QTY): IVL_INT\r\n    IVL_MO(any: MO, high: MO,
highClosed: Boolean, low: MO, lowClosed: Boolean, width: QTY):
IVL_MO\r\n    IVL_PQ(any: PQ, high: PQ, highClosed: Boolean, low: PQ,
lowClosed: Boolean, width: QTY): IVL_PQ\r\n    IVL_QTY(any: QTY, high:
QTY, highClosed: Boolean, low: QTY, lowClosed: Boolean, width: QTY):
IVL_QTY\r\n    IVL_REAL(any: REAL, high: REAL, highClosed: Boolean, low:
REAL, lowClosed: Boolean, width: QTY): IVL_REAL\r\n    IVL_TS(any: TS,
high: TS, highClosed: Boolean, low: TS, lowClosed: Boolean, width: QTY):
IVL_TS\r\n    LIST_AD(value: Sequence(AD)): LIST_AD\r\n    LIST_BL(value:
Sequence(BL)): LIST_BL\r\n    LIST_CD(value: Sequence(CD)): LIST_CD\r\n
LIST_CO(value: Sequence(CO)): LIST_CO\r\n    LIST_CS(value: Sequence(CS)):
LIST_CS\r\n    LIST_ED(value: Sequence(ED)): LIST_ED\r\n    LIST_EN(value:
Sequence(EN)): LIST_EN\r\n    LIST_II(value: Sequence(II)): LIST_II\r\n
LIST_INT(value: Sequence(INT)): LIST_INT\r\n    LIST_MO(value: Sequence
(MO)): LIST_MO\r\n    LIST_PQ(value: Sequence(PQ)): LIST_PQ\r\n
LIST_REAL(value: Sequence(REAL)): LIST_REAL\r\n    LIST_RTO(value: Sequence
(RTO)): LIST_RTO\r\n    LIST_SC(value: Sequence(SC)): LIST_SC\r\n
LIST_ST(value: Sequence(ST)): LIST_ST\r\n    LIST_TEL(value: Sequence
(TEL)): LIST_TEL\r\n    LIST_TS(value: Sequence(TS)): LIST_TS\r\n    MO
(currency: String, value: Real): MO\r\n    NPPD_AD(value: Sequence
(UVP_AD)): NPPD_AD\r\n    NPPD_BL(value: Sequence(UVP_BL)): NPPD_BL\r\n
NPPD_CD(value: Sequence(UVP_CD)): NPPD_CD\r\n    NPPD_CO(value: Sequence
(UVP_CO)): NPPD_CO\r\n    NPPD_CS(value: Sequence(UVP_CS)): NPPD_CS\r\n
NPPD_ED(value: Sequence(UVP_ED)): NPPD_ED\r\n    NPPD_EN(value: Sequence
(UVP_EN)): NPPD_EN\r\n    NPPD_II(value: Sequence(UVP_II)): NPPD_II\r\n
NPPD_INT(value: Sequence(UVP_INT)): NPPD_INT\r\n    NPPD_MO(value: Sequence
(UVP_MO)): NPPD_MO\r\n    NPPD_PQ(value: Sequence(UVP_PQ)): NPPD_PQ\r\n
NPPD_REAL(value: Sequence(UVP_REAL)): NPPD_REAL\r\n    NPPD_RTO(value:
Sequence(UVP_RTO)): NPPD_RTO\r\n    NPPD_SC(value: Sequence(UVP_SC)):
NPPD_SC\r\n    NPPD_ST(value: Sequence(UVP_ST)): NPPD_ST\r\n    NPPD_TEL
(value: Sequence(UVP_TEL)): NPPD_TEL\r\n    NPPD_TS(value: Sequence
(UVP_TS)): NPPD_TS\r\n    PIVL_TS(alignment: String, count: INT,
frequency: RTO, isFlexible: Boolean, period: PQ, phase: IVL_TS):
PIVL_TS\r\n    PQ(value: Real, _unit_: String): PQ\r\n    PQR(value:
Real): PQR\r\n    QSC_TS(code: CD): QSC_TS\r\n    QSD_PQ(minuend: QSET_PQ,
subtrahend: QSET_PQ): QSD_PQ\r\n    QSD_TS(minuend: QSET_TS, subtrahend:
QSET_TS): QSD_TS\r\n    QSET_CO(originalText: ED): QSET_CO\r\n    QSET_INT
(originalText: ED): QSET_INT\r\n    QSET_MO(originalText: ED):
QSET_MO\r\n    QSET_PQ(originalText: ED): QSET_PQ\r\n    QSET_QTY
(originalText: ED): QSET_QTY\r\n    QSET_REAL(originalText: ED):
QSET_REAL\r\n    QSET_TS(originalText: ED): QSET_TS\r\n    QSI_PQ(term:
Sequence(QSET_PQ)): QSI_PQ\r\n    QSI_TS(term: Sequence(QSET_TS)):
QSI_TS\r\n    QSP_PQ(high: QSET_PQ, low: QSET_PQ): QSP_PQ\r\n    QSP_TS
(high: QSET_TS, low: QSET_TS): QSP_TS\r\n    QSS_PQ(term: Sequence(PQ)):
QSS_PQ\r\n    QSS_TS(term: Sequence(TS)): QSS_TS\r\n    QSU_PQ(term:
Sequence(QSET_PQ)): QSU_PQ\r\n    QSU_TS(term: Sequence(QSET_TS)):
QSU_TS\r\n    QTY(expression: ED, originalText: ED, uncertainRange:
IVL_QTY, uncertainty: QTY, uncertaintyType: String): QTY\r\n    REAL
(value: Real): REAL\r\n    RTO(denominator: QTY, numerator: QTY):
RTO\r\n    SC(code: CD): SC\r\n    SD_TEXT(ID: String, language: String,
mediaType: String, styleCode: Set(Code)): SD_TEXT\r\n    SD_TITLE(ID:
String, language: String, mediaType: String, styleCode: Set(Code)):
SD_TITLE\r\n    set_Code(value: Set(Code)): Set(Code)\r\n
```

```
set_CodingRationale(value: Set(CodingRationale)): Set(CodingRationale)
\r\n    set_EntityNamePartQualifier(value: Set(EntityNamePartQualifier)):
Set(EntityNamePartQualifier)\r\n    set_EntityNameUse(value: Set
(EntityNameUse)): Set(EntityNameUse)\r\n    set_TelecommunicationAddressUse
(value: Set(TelecommunicationAddressUse)): Set(TelecommunicationAddressUse)
\r\n    set_TelecommunicationCapability(value: Set
(TelecommunicationCapability)): Set(TelecommunicationCapability)\r\n
SLIST_INT(digit: Sequence(INT), origin: INT, scale: QTY): SLIST_INT\r\n
SLIST_PQ(digit: Sequence(INT), origin: PQ, scale: QTY): SLIST_PQ\r\n
SLIST_REAL(digit: Sequence(INT), origin: REAL, scale: QTY):
SLIST_REAL\r\n    SLIST_TS(digit: Sequence(INT), origin: TS, scale: QTY):
SLIST_TS\r\n    ST(language: String, translation: Sequence(ST), value:
String): ST\r\n    TEL(capabilities: Set(TelecommunicationCapability),
use: Set(TelecommunicationAddressUse), useablePeriod: QSET_TS, value:
String): TEL\r\n    TS(value: String): TS\r\n    Uid(ancestor: String):
String\r\n    Uri(ancestor: String): String\r\n    UVP_AD(probability:
Real, value: AD): UVP_AD\r\n    UVP_BL(probability: Real, value: BL):
UVP_BL\r\n    UVP_CD(probability: Real, value: CD): UVP_CD\r\n    UVP_CO
(probability: Real, value: CO): UVP_CO\r\n    UVP_CS(probability: Real,
value: CS): UVP_CS\r\n    UVP_ED(probability: Real, value: ED):
UVP_ED\r\n    UVP_EN(probability: Real, value: EN): UVP_EN\r\n    UVP_II
(probability: Real, value: II): UVP_II\r\n    UVP_INT(probability: Real,
value: INT): UVP_INT\r\n    UVP_MO(probability: Real, value: MO):
UVP_MO\r\n    UVP_PQ(probability: Real, value: PQ): UVP_PQ\r\n    UVP_REAL
(probability: Real, value: REAL): UVP_REAL\r\n    UVP_RTO(probability:
Real, value: RTO): UVP_RTO\r\n    UVP_SC(probability: Real, value: SC):
UVP_SC\r\n    UVP_ST(probability: Real, value: ST): UVP_ST\r\n    UVP_TEL
(probability: Real, value: TEL): UVP_TEL\r\n    UVP_TS(probability: Real,
value: TS): UVP_TS\r\n    XP(code: String, codeSystem: String,
codeSystemVersion: String, language: String, nullFlavor: NullFlavor,
value: String): XP\r\n    XReference(xref: String): XReference\r\n\r\n
class GLIST_PQ extends ANY\r\n    denominator: Integer\r\n    head:
PQ\r\n    increment: QTY\r\n    period: Integer\r\n\r\n  class GLIST_REAL
extends ANY\r\n    denominator: Integer\r\n    head: REAL\r\n
increment: QTY\r\n    period: Integer\r\n\r\n  class GLIST_TS extends
ANY\r\n    denominator: Integer\r\n    head: TS\r\n    increment:
QTY\r\n    period: Integer\r\n\r\n  class HD extends Any\r\n
NamespaceID: String\r\n    UniversalID: String\r\n    UniversalIDType:
String\r\n\r\n  class LIST_AD extends LIST_T\r\n    item: Sequence(AD)
\r\n\r\n  class LIST_BL extends LIST_T\r\n    item: Sequence(BL)\r\n\r\n
class LIST_CD extends LIST_T\r\n    item: Sequence(CD)\r\n\r\n  class
LIST_CO extends LIST_T\r\n    item: Sequence(CO)\r\n\r\n  class LIST_CS
extends LIST_T\r\n    item: Sequence(CS)\r\n\r\n  class LIST_ED extends
LIST_T\r\n    item: Sequence(ED)\r\n\r\n  class LIST_EN extends
LIST_T\r\n    item: Sequence(EN)\r\n\r\n  class LIST_II extends
LIST_T\r\n    item: Sequence(II)\r\n\r\n  class LIST_INT extends
LIST_T\r\n    item: Sequence(INT)\r\n\r\n  class LIST_MO extends
LIST_T\r\n    item: Sequence(MO)\r\n\r\n  class LIST_PQ extends
LIST_T\r\n    item: Sequence(PQ)\r\n\r\n  class LIST_REAL extends
LIST_T\r\n    item: Sequence(REAL)\r\n\r\n  class LIST_RTO extends
LIST_T\r\n    item: Sequence(RTO)\r\n\r\n  class LIST_SC extends
LIST_T\r\n    item: Sequence(SC)\r\n\r\n  class LIST_ST extends
LIST_T\r\n    item: Sequence(ST)\r\n\r\n  class LIST_TEL extends
LIST_T\r\n    item: Sequence(TEL)\r\n\r\n  class LIST_TS extends
LIST_T\r\n    item: Sequence(TS)\r\n\r\n  class AD extends ANY\r\n
isNotOrdered: Boolean\r\n    part: Sequence(ADXP)\r\n    use: Set
(PostalAddressUse)\r\n    useablePeriod: QSET_TS\r\n    asXAD:
XAD\r\n\r\n  class II extends ANY\r\n    displayable: Boolean\r\n
extension: String\r\n    identifierName: String\r\n    reliability:
String\r\n    root: String\r\n    scope: String\r\n    asHD: HD\r\n\r\n
class INT extends QTY\r\n    value: Integer\r\n\r\n    -- methods\r\n    +
(other: INT): INT\r\n    -(other: INT): INT\r\n    -(): INT\r\n    *
(other: INT): INT\r\n    /(other: INT): REAL\r\n    /(other: REAL):
REAL\r\n    abs(): INT\r\n    comparable(other: QTY): BL\r\n    div(other:
INT): INT\r\n    mod(other: INT): INT\r\n    min(other: INT): INT\r\n
max(other: INT): INT\r\n    >=(other: INT): Boolean\r\n    >=(other:
Integer): Boolean\r\n    >(other: INT): Boolean\r\n    >(other: Integer):
Boolean\r\n    <=(other: INT): Boolean\r\n    <=(other: Integer):
Boolean\r\n    <(other: INT): Boolean\r\n    <(other: Integer):
Boolean\r\n\r\n  class QSET_CO extends ANY\r\n    originalText:
ED\r\n\r\n  class QSET_INT extends ANY\r\n    originalText: ED\r\n\r\n
```

```
class QSET_MO extends ANY\r\n    originalText: ED\r\n\r\n  class QSET_PQ
extends ANY\r\n    originalText: ED\r\n\r\n  class QSET_QTY extends
ANY\r\n    originalText: ED\r\n\r\n  class QSET_REAL extends ANY\r\n
originalText: ED\r\n\r\n  class IVL_TS extends QSET_TS\r\n    low:
TS\r\n    high: TS\r\n    width: QTY\r\n    any: TS\r\n    lowClosed:
Boolean\r\n    highClosed: Boolean\r\n\r\n    -- methods\r\n    contains
(x: TS): BL\r\n\r\n  class MO extends QTY\r\n    currency: String\r\n
value: Real\r\n\r\n    -- methods\r\n    +(other: MO): MO\r\n    -(other:
MO): MO\r\n    *(other: REAL): MO\r\n    /(other: REAL): MO\r\n    min
(other: MO): MO\r\n    max(other: MO): MO\r\n\r\n  class NPPD_AD extends
ANY\r\n    value: Sequence(UVP_AD)\r\n\r\n  class NPPD_BL extends
ANY\r\n    value: Sequence(UVP_BL)\r\n\r\n  class NPPD_CD extends
ANY\r\n    value: Sequence(UVP_CD)\r\n\r\n  class NPPD_CO extends
ANY\r\n    value: Sequence(UVP_CO)\r\n\r\n  class NPPD_CS extends
ANY\r\n    value: Sequence(UVP_CS)\r\n\r\n  class NPPD_ED extends
ANY\r\n    value: Sequence(UVP_ED)\r\n\r\n  class NPPD_EN extends
ANY\r\n    value: Sequence(UVP_EN)\r\n\r\n  class NPPD_II extends
ANY\r\n    value: Sequence(UVP_II)\r\n\r\n  class NPPD_INT extends
ANY\r\n    value: Sequence(UVP_INT)\r\n\r\n  class NPPD_MO extends
ANY\r\n    value: Sequence(UVP_MO)\r\n\r\n  class NPPD_PQ extends
ANY\r\n    value: Sequence(UVP_PQ)\r\n\r\n  class NPPD_REAL extends
ANY\r\n    value: Sequence(UVP_REAL)\r\n\r\n  class NPPD_RTO extends
ANY\r\n    value: Sequence(UVP_RTO)\r\n\r\n  class NPPD_SC extends
ANY\r\n    value: Sequence(UVP_SC)\r\n\r\n  class NPPD_ST extends
ANY\r\n    value: Sequence(UVP_ST)\r\n\r\n  class NPPD_TEL extends
ANY\r\n    value: Sequence(UVP_TEL)\r\n\r\n  class NPPD_TS extends
ANY\r\n    value: Sequence(UVP_TS)\r\n\r\n  class PIVL_TS extends
QSET_TS\r\n    alignment: String\r\n    count: INT\r\n    frequency:
RTO\r\n    isFlexible: Boolean\r\n    period: PQ\r\n    phase:
IVL_TS\r\n\r\n  class PQ extends QTY\r\n    codingRationale: String\r\n
translation: Sequence(PQR)\r\n    value: Real\r\n    unit:
String\r\n\r\n    -- methods\r\n    abs(): PQ\r\n    convert(_unit_:
String): PQ\r\n    canonical(): PQ\r\n    inverted(): PQ\r\n    max(other:
PQ): PQ\r\n    min(other: PQ): PQ\r\n    toInterval(): IVL_PQ\r\n    /
(other: Integer): PQ\r\n    /(other: Real): PQ\r\n    /(other: PQ):
PQ\r\n    >=(other: PQ): Boolean\r\n    >(other: PQ): Boolean\r\n    <=
(other: PQ): Boolean\r\n    <(other: PQ): Boolean\r\n    -(): PQ\r\n    -
(other: PQ): PQ\r\n    *(other: Integer): PQ\r\n    *(other: Real):
PQ\r\n    *(other: PQ): PQ\r\n    +(other: PQ): PQ\r\n\r\n  class PQR
extends CD\r\n    value: Real\r\n\r\n  class QSC_TS extends QSET_TS\r\n
code: CD\r\n\r\n  class QSD_PQ extends QSET_PQ\r\n    minuend:
QSET_PQ\r\n    subtrahend: QSET_PQ\r\n\r\n  class QSD_TS extends
QSET_TS\r\n    minuend: QSET_TS\r\n    subtrahend: QSET_TS\r\n\r\n  class
IVL_CO extends QSET_CO\r\n    low: CO\r\n    high: CO\r\n    width:
QTY\r\n    any: CO\r\n    lowClosed: Boolean\r\n    highClosed:
Boolean\r\n\r\n  class IVL_INT extends QSET_INT\r\n    low: INT\r\n
high: INT\r\n    width: QTY\r\n    any: INT\r\n    lowClosed:
Boolean\r\n    highClosed: Boolean\r\n\r\n  class IVL_MO extends
QSET_MO\r\n    low: MO\r\n    high: MO\r\n    width: QTY\r\n    any:
MO\r\n    lowClosed: Boolean\r\n    highClosed: Boolean\r\n\r\n  class
IVL_PQ extends QSET_PQ\r\n    low: PQ\r\n    high: PQ\r\n    width:
QTY\r\n    any: PQ\r\n    lowClosed: Boolean\r\n    highClosed:
Boolean\r\n\r\n    -- methods\r\n    contains(x: PQ): BL\r\n    contains
(x: IVL_PQ): BL\r\n\r\n  class IVL_QTY extends QSET_QTY\r\n    low:
QTY\r\n    high: QTY\r\n    width: QTY\r\n    any: QTY\r\n    lowClosed:
Boolean\r\n    highClosed: Boolean\r\n\r\n    -- methods\r\n    contains
(x: IVL_PQ): BL\r\n    contains(x: IVL_QTY): BL\r\n    contains(x: QTY):
BL\r\n\r\n  class IVL_REAL extends QSET_REAL\r\n    low: REAL\r\n    high:
REAL\r\n    width: QTY\r\n    any: REAL\r\n    lowClosed: Boolean\r\n
highClosed: Boolean\r\n\r\n  class EIVL_TS extends QSET_TS\r\n    event:
String\r\n    offset: IVL_PQ\r\n\r\n  class QSI_PQ extends QSET_PQ\r\n
term: Sequence(QSET_PQ)\r\n\r\n  class QSI_TS extends QSET_TS\r\n    term:
Sequence(QSET_TS)\r\n\r\n  class QSP_PQ extends QSET_PQ\r\n    high:
QSET_PQ\r\n    low: QSET_PQ\r\n\r\n  class QSP_TS extends QSET_TS\r\n
high: QSET_TS\r\n    low: QSET_TS\r\n\r\n  class QSS_PQ extends
QSET_PQ\r\n    term: Sequence(PQ)\r\n\r\n  class QSS_TS extends
QSET_TS\r\n    term: Sequence(TS)\r\n\r\n  class QSU_PQ extends
QSET_PQ\r\n    term: Sequence(QSET_PQ)\r\n\r\n  class QSU_TS extends
QSET_TS\r\n    term: Sequence(QSET_TS)\r\n\r\n  class CO extends
QTY\r\n    code: CD\r\n    value: Real\r\n\r\n    -- methods\r\n    +
(other: CO): CO\r\n    -(other: CO): CO\r\n    min(other: CO): CO\r\n
```

```
max(other: CO): CO\r\n\r\n  class REAL extends QTY\r\n    value:
Real\r\n\r\n    -- methods\r\n    +(other: REAL): REAL\r\n    +(other:
Real): REAL\r\n    +(other: INT): REAL\r\n    +(other: Integer):
REAL\r\n    -(other: REAL): REAL\r\n    -(other: Real): REAL\r\n    -
(other: INT): REAL\r\n    -(other: Integer): REAL\r\n    *(other: REAL):
REAL\r\n    *(other: Real): REAL\r\n    *(other: INT): REAL\r\n    *
(other: Integer): REAL\r\n    /(other: REAL): REAL\r\n    /(other: Real):
REAL\r\n    /(other: INT): REAL\r\n    /(other: Integer): REAL\r\n    min
(other: REAL): REAL\r\n    min(other: INT): REAL\r\n    max(other: REAL):
REAL\r\n    max(other: INT): REAL\r\n    -(): REAL\r\n    <(other: REAL):
Boolean\r\n    <(other: Real): Boolean\r\n    <(other: INT):
Boolean\r\n    <(other: Integer): Boolean\r\n    <=(other: REAL):
Boolean\r\n    <=(other: Real): Boolean\r\n    <=(other: INT):
Boolean\r\n    <=(other: Integer): Boolean\r\n    >=(other: REAL):
Boolean\r\n    >=(other: Real): Boolean\r\n    >=(other: INT):
Boolean\r\n    >=(other: Integer): Boolean\r\n    >(other: REAL):
Boolean\r\n    >(other: Real): Boolean\r\n    >(other: INT):
Boolean\r\n    >(other: Integer): Boolean\r\n    abs(): REAL\r\n
ceiling(): INT\r\n    floor(): INT\r\n    round(): INT\r\n    inverted():
REAL\r\n    power(other: REAL): REAL\r\n    toInterval(): IVL_REAL\r\n
comparable(other: QTY): BL\r\n\r\n  class RTO extends QTY\r\n
numerator: QTY\r\n    denominator: QTY\r\n\r\n  class ST extends
ANY\r\n    language: String\r\n    translation: Sequence(ST)\r\n    value:
String\r\n\r\n    -- methods\r\n    mediaType(): String\r\n    size():
Integer\r\n    concat(other: ST): ST\r\n    substring(lower: INT, upper:
INT): ST\r\n    toInteger(): INT\r\n    toReal(): REAL\r\n
toTimestamp(): TS\r\n\r\n  class SD_TEXT extends ANY\r\n    ID:
String\r\n    language: String\r\n    mediaType: String\r\n    styleCode:
Set(Code)\r\n\r\n  class SD_TITLE extends ANY\r\n    ID: String\r\n
language: String\r\n    mediaType: String\r\n    styleCode: Set(Code)
\r\n\r\n  class SLIST_INT extends ANY\r\n    digit: Sequence(INT)\r\n
origin: INT\r\n    scale: QTY\r\n\r\n  class SLIST_PQ extends ANY\r\n
digit: Sequence(INT)\r\n    origin: PQ\r\n    scale: QTY\r\n\r\n  class
SLIST_REAL extends ANY\r\n    digit: Sequence(INT)\r\n    origin:
REAL\r\n    scale: QTY\r\n\r\n  class SLIST_TS extends ANY\r\n    digit:
Sequence(INT)\r\n    origin: TS\r\n    scale: QTY\r\n\r\n  class SC
extends ST\r\n    code: CD\r\n\r\n  class TEL extends ANY\r\n
capabilities: Set(TelecommunicationCapability)\r\n    use: Set
(TelecommunicationAddressUse)\r\n    useablePeriod: QSET_TS\r\n    value:
String\r\n\r\n    -- methods\r\n    canonical(): TEL\r\n\r\n  class TS
extends QTY\r\n    value: String\r\n\r\n    -- methods\r\n    precision():
Integer\r\n    +(other: PQ): TS\r\n    -(other: PQ): TS\r\n    -(other:
TS): PQ\r\n    min(other: TS): TS\r\n    max(other: TS): TS\r\n
toInterval(): IVL_TS\r\n    <(other: TS): Boolean\r\n    <=(other: TS):
Boolean\r\n    >=(other: TS): Boolean\r\n    >(other: TS): Boolean\r\n
equals(other: TS): BL\r\n\r\n  class UVP_AD extends ANY\r\n
probability: Real\r\n    value: AD\r\n\r\n  class UVP_BL extends
ANY\r\n    probability: Real\r\n    value: BL\r\n\r\n  class UVP_CD
extends ANY\r\n    probability: Real\r\n    value: CD\r\n\r\n  class
UVP_CO extends ANY\r\n    probability: Real\r\n    value: CO\r\n\r\n
class UVP_CS extends ANY\r\n    probability: Real\r\n    value:
CS\r\n\r\n  class UVP_ED extends ANY\r\n    probability: Real\r\n
value: ED\r\n\r\n  class UVP_EN extends ANY\r\n    probability:
Real\r\n    value: EN\r\n\r\n  class UVP_II extends ANY\r\n
probability: Real\r\n    value: II\r\n\r\n  class UVP_INT extends
ANY\r\n    probability: Real\r\n    value: INT\r\n\r\n  class UVP_MO
extends ANY\r\n    probability: Real\r\n    value: MO\r\n\r\n  class
UVP_PQ extends ANY\r\n    probability: Real\r\n    value: PQ\r\n\r\n
class UVP_REAL extends ANY\r\n    probability: Real\r\n    value:
REAL\r\n\r\n  class UVP_RTO extends ANY\r\n    probability: Real\r\n
value: RTO\r\n\r\n  class UVP_SC extends ANY\r\n    probability:
Real\r\n    value: SC\r\n\r\n  class UVP_ST extends ANY\r\n
probability: Real\r\n    value: ST\r\n\r\n  class UVP_TEL extends
ANY\r\n    probability: Real\r\n    value: TEL\r\n\r\n  class UVP_TS
extends ANY\r\n    probability: Real\r\n    value: TS\r\n\r\n  class XAD
extends Any\r\n    StreetAddress: String\r\n    OtherDesignation:
String\r\n    City: String\r\n    State: String\r\n    PostalCode:
String\r\n    Country: String\r\n    AddressType: String\r\n\r\n  class
XPN extends Any\r\n    FamilyName: String\r\n    GivenName: String\r\n
SecondName: String\r\n    Suffix: String\r\n    Prefix: String\r\n
Degree: String\r\n    NameTypeCode: String\r\n\r\n  class ADXP extends
```

```
XP\r\n    type: AddressPartType\r\n\r\n  class XCN extends XPN\r\n
IDNumber: String\r\n    AssigningAuthority: HD\r\n\r\n  class XReference
extends Any\r\n    xref: String\r\n\r\n  class AddressPartType Enum(\"
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\", \"DPID\")
\r\n\r\n  class PostalAddressUse Enum(\"H\", \"HP\", \"HV\", \"WP\", \"
DIR\", \"PUB\", \"BAD\", \"PHYS\", \"PST\", \"TMP\", \"ABC\", \"IDE\", \"
SYL\", \"SRCH\", \"SNDX\", \"PHON\")\r\n\r\n  class CE extends CD\r\n\r\n
class CV extends CE\r\n\r\n\r\nEndPackage\r\n"
  }
 ]
}
```