Data types

Text

- $xsd{:}\ string$
- Text in menschlicher Sprache

```
anychar = %x09-10FFFF ; but not banned, as with all ABNF rules
Text    = *anychar
```

Integer

- $xsd{:}\ nonNegativeInteger$
- Nicht-leere Sequenz von ASCII Dezimalzahlen

```
Integer = 1*digit
```

Enumeration

- $g7{:}\ type\text{-}Enum$
- String-Repräsentation einer Auswahl von Optionen

```
Enum    = Tag
```

Date

- Datentyp um Datum zu speichern

```
DateValue    = date / DatePeriod / dateRange / dateApprox / ""
DateExact    = day D month D year  ; in Gregorian calendar
DatePeriod   = %s"FROM" D date [D %s"TO" D date]
             / %s"TO" D date
             / ""

date         = [calendar D] [[day D] month D] year [D epoch]
dateRange    = %s"BET" D date D %s"AND" D date
             / %s"AFT" D date
             / %s"BEF" D date
dateApprox   = (%s"ABT" / %s"CAL" / %s"EST") D date

dateRestrict = %s"FROM" / %s"TO" / %s"BET" / %s"AND" / %s"BEF"
             / %s"AFT" / %s"ABT" / %s"CAL" / %s"EST" / %s"BCE"

calendar = %s"GREGORIAN" / %s"JULIAN" / %s"FRENCH_R" / %s"HEBREW"
         / extTag

day      = Integer
year     = Integer
month    = stdTag / extTag  ; constrained by calendar
epoch    = %s"BCE" / extTag ; constrained by calendar
```

Time

- Uhrzeit auf einer 24-Stunden Uhr

```
Time     =  hour ":" minute [":" second ["." fraction]] [%s"Z"]

hour     = digit / ("0" / "1") digit / "2" ("0" / "1" / "2" / "3")
minute   = ("0" / "1" / "2" / "3" / "4" / "5") digit
second   = ("0" / "1" / "2" / "3" / "4" / "5") digit
fraction = 1*digit
```

Age

- $g7{:}\,type\text{-}\mathrm{Age}$
- Alter repräsentiert in Anzahl der Jahre, Monate, Wochen und Tagen

```
Age         = [ageBound D] ageDuration

ageBound    = "<" / ">"
ageDuration = years [D months] [D weeks] [D days]
            / months [D weeks] [D days]
            / weeks [D days]
            / days

years   = Integer %x79    ; 35y
months  = Integer %x6D    ; 11m
weeks   = Integer %x77    ; 8w
days    = Integer %x64    ; 21d
```

List

- $g7{:}\,type\text{-}List\#Text$
- $g7{:}\,type\text{-}List\#Enmu$

```
List      = listItem *(listDelim listItem)
listItem  = "" / nocommasp / nocommasp *nocomma nocommasp
listDelim = *D "," *D
nocomma   = %x09-2B / %x2D-10FFFF
nocommasp = %x09-1D / %x21-2B / %x2D-10FFFF

List-Text = List
List-Enum = Enum *(listDelim Enum)
```

Personal Name

- $g7{:}\,type - Name$

```
PersonalName = nameStr
             / [nameStr] "/" [nameStr] "/" [nameStr]

nameChar     = %x20-2E / %x30-10FFFF  ; any but '/' and '\t'
nameStr      = 1*nameChar
```

Language

- $xsd{:}\,Language$
- Menschliche oder familienzugehörige Sprachen

Media type	• Kodierte Informationen in Bytes oder Character
	
	`MediaType = mt-type "/" mt-subtype *(";" mt-parameter)`
	
	`mt-type = mt-token`
	`mt-subtype = mt-token`
	`mt-parameter = mt-attribute "=" mt-value`
	`mt-token = 1*mt-char`
	`mt-attribute = mt-token`
	`mt-value = mt-token / quoted-string`
	`mt-char = %x20-21 / %x23-27 / %x2A-2B / %x2D-2E ; not "(),/`
	`/ %x30-39 / %x41-5A / %x5E-7E ; not :;<=>?@[\]`
	
	`mt-qstring = %x22 *(mt-qtext / mt-qpair) %x22`
	`mt-qtext = %x09-0A / %x20-21 / %x23-5B / %x5D-7E ; not CR "\`
	`mt-qpair = "\" %x09-7E`

Special	• $xsd{:}\ string$ • Spezial-Datentyp für bestimmte Anwendungsfälle → ist unique für jede structure `Special = Text`