JSON Representation of DICOM Structured Reports

DICOM WG 23

David Clunie

Trial Use Phase

2020/01/16

http://medium.com/adhive/disruptive-ai-controlled-advertising-cd90a07452cb

Annotation interoperability matters now

- Previously:
  - little incentive to annotate
  - few tools to create or view annotations
  - annotation interoperability was a low priority for product managers
  - presentation rather than semantics were the priority for annotation tools
- Now:
  - semantic annotations have (real monetary) value beyond primary use case
  - recognition of existence of unanticipated re-use cases
  - annotations are expensive to create/recreate retrospectively
  - more expensive to process if proprietary rather than OTS standard
  - AI-generated annotations need to be interoperable for display
  - “interactive” AI requires interoperable annotation exchange
  - AI vendors unlikely to be the same as scanner/PACS vendors – mix and match

DICOM SR and AI

- DICOM SR is a generic solution for:
  - fundamental encoding of measurements, categorical results, using codes and referencing images, waveforms as well as spatial and temporal coordinates
  - reusable sub-templates for specific scenarios that are common to different use cases and applications
  - generic root level templates for non-specific measurements (e.g., TID 1500)
  - linking other objects related to results and measurements (such as SEG, Parametric Map and RWVM)
- Specific templates for:
  - traditional CAD applications that are relevant to AI
  - traditional human operator measurements that may now be made by AI

DICOM SR and the developer

- Traditional DICOM SR encoding requires use of a toolkit and an API with a non-trivial learning curve (binary encoding intractable by hand)
- AI algorithm developer may not need to know about the “composite context” (patient/study/series +/- workflow metadata) of the encounter
- Impedance mismatch between
  - PACS-orientated “DICOM image in, DICOM SEG + SR out”
  - Algorithm-developer orientated “PNG in, PNG + JSON out”
- Even XML is deemed excessive/too complicated by AI developer community
- DICOMweb JSON encoding is also intractable for SR, since it is hexadecimal tag, individual data element orientated (no SR content item abstraction)

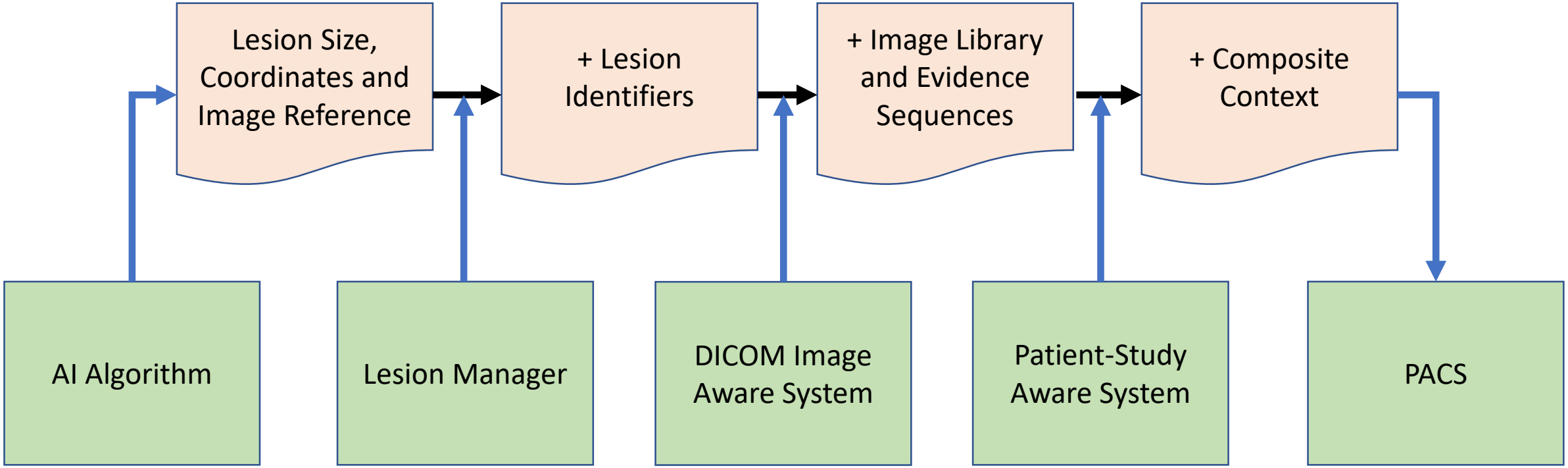
Goals for Simplified DICOM SR in JSON

- Full-fidelity round trip with actual DICOM SR for all constructs (any template)
- Simple (enough to hand write or copy from examples)
- Compact (even terse)
- Understandable (relatively)
- Unambiguous (easily parsable)
- Leverage any existing actual or de facto JSON or evolving AI standards
- Platform independent
- Capable of encoding extracts separated from composite context (such as without “header” rather than content tree, image library, etc., which could be added by separate tool/pass)

Non-Goals for Simplified DICOM SR in JSON

- Not an alternative/competitor to existing PS3.18 Annex F JSON for non-SR objects
- Not an alternative/competing persistent form to be serialized and stored, as opposed to binary DICOM SR stored in PACS/VNA
- Not an abstraction of template-specific concepts or alternative information models for similar content
- No template-specific constraints or optimizations
- Not a means for defining a new validation mechanism for SR content (template-defined), but does not prohibit it

Pipeline to add missing stuff to JSON



Design Decisions – Business Names

- No hexadecimal numbers for “header” attributes – leverage DICOMweb JSON encoding but with PS3.6 keywords rather than numeric tags
- Abstract the content items (i.e., name-value pairs), as if they were attributes, rather than exposing their component attributes
- No obscure alphanumeric codes in content tree – use “business names” concept from Green CDA (not dissimilar to JSON-LD)
- Codes are defined in separate “business names” JSON file that acts as a dictionary – do not need to be standardized (but may be in future, like keywords)

Design Decisions – JSON Structure

- Use JSON Objects where identity is important but not order
- Use business name as name of JSON Object's name-value pair
- Use JSON Arrays to preserve order
- Use JSON Arrays to allow sibling JSON Objects with same name
- Use a JSON Array to encode children
- Collapse unnecessary JSON Arrays into single value when possible for business names and top level data elements
- Omit data element VR if it can be found in dictionary or business name file
- Omit explicit value type and relationship type if they can be deduced from context, or defined in the separate business names file
- Add annotations (specific object names starting with "_" symbol) to content item specific attributes, and to provide target and source for by-reference relationships
- Use keywords for well known UIDs, e.g., Storage SOP Classes

Example 1 – hexdump of the original (partial)

```
00000390  20 00 11 00 49 53 04 00  34 35 37 38 20 00 13 00  | ...IS..4578 ...|
000003a0  49 53 02 00 31 20 40 00  40 a0 43 53 0a 00 43 4f  |IS..1 @.@.CS..CO|
000003b0  4e 54 41 49 4e 45 52 20  40 00 43 a0 53 51 00 00  |NTAINER @.C.SQ..|
000003c0  ff ff ff ff fe ff 00 e0  ff ff ff ff 08 00 00 01  |................|
000003d0  53 48 06 00 31 32 36 30  30 30 08 00 02 01 53 48  |SH..126000....SH|
000003e0  04 00 44 43 4d 20 08 00  04 01 4c 4f 1a 00 49 6d  |..DCM ....LO..Im|
000003f0  61 67 69 6e 67 20 4d 65  61 73 75 72 65 6d 65 6e  |aging Measuremen|
00000400  74 20 52 65 70 6f 72 74  fe ff 0d e0 00 00 00 00  |t Report........|
00000410  fe ff dd e0 00 00 00 00  40 00 50 a0 43 53 08 00  |........@.P.CS..|
00000420  53 45 50 41 52 41 54 45  40 00 78 a0 53 51 00 00  |SEPARATE@.x.SQ..|
00000430  ff ff ff ff fe ff 00 e0  ff ff ff ff 08 00 80 00  |................|
00000440  4c 4f 00 00 08 00 82 00  53 51 00 00 ff ff ff ff  |LO......SQ......|
00000450  fe ff dd e0 00 00 00 00  40 00 01 11 53 51 00 00  |........@...SQ..|
00000460  ff ff ff ff fe ff dd e0  00 00 00 00 40 00 84 a0  |............@...|
00000470  43 53 04 00 50 53 4e 20  40 00 23 a1 50 4e 14 00  |CS..PSN @.#.PN..|
00000480  61 63 63 6f 6d 70 6c 69  73 68 65 64 5f 70 65 61  |accomplished_pea|
00000490  66 6f 77 6c fe ff 0d e0  00 00 00 00 fe ff dd e0  |fowl............|
000004a0  00 00 00 00 40 00 72 a3  53 51 00 00 ff ff ff ff  |....@.r.SQ......|
000004b0  fe ff dd e0 00 00 00 00  40 00 75 a3 53 51 00 00  |........@.u.SQ..|
000004c0  ff ff ff ff fe ff 00 e0  ff ff ff ff 08 00 15 11  |................|
000004d0  53 51 00 00 ff ff ff ff  fe ff 00 e0 ff ff ff ff  |SQ..............|
000004e0  08 00 99 11 53 51 00 00  ff ff ff ff fe ff 00 e0  |....SQ..........|
000004f0  ff ff ff ff 08 00 50 11  55 49 1a 00 31 2e 32 2e  |......P.UI..1.2.|
00000500  38 34 30 2e 31 30 30 30  38 2e 35 2e 31 2e 34 2e  |840.10008.5.1.4.|
00000510  31 2e 31 2e 32 00 08 00  55 11 55 49 40 00 31 2e  |1.1.2...U.UI@.1.|
00000520  33 2e 36 2e 31 2e 34 2e  31 2e 31 34 35 31 39 2e  |3.6.1.4.1.14519.|
00000530  35 2e 32 2e 31 2e 39 32  30 33 2e 34 30 30 34 2e  |5.2.1.9203.4004.|
00000540  32 36 38 30 31 38 34 32  32 32 38 38 38 31 38 35  |2680184222888185|
00000550  37 33 32 32 36 35 31 36  30 32 33 37 36 32 fe ff  |73226516023762..|
```

Example 1 – dcsrump of the original

```
: CONTAINER: (126000,DCM,"Imaging Measurement Report")  [SEPARATE] (DCMR,1500)
       >CONTAINS: CONTAINER: (126010,DCM,"Imaging Measurements")  [SEPARATE]
              >>CONTAINS: CONTAINER: (125007,DCM,"Measurement Group")  [SEPARATE]
                     >>>CONTAINS: NUM: (410668003,SCT,"Length")  = 97.08595644 (mm,UCUM,"mm")
                            >>>>INFERRED FROM: SCOORD: (121055,DCM,"Path")  = POLYLINE {186.41325378418,274.590057373047,89.1049728393555,374.727081298828}
                                   >>>>>SELECTED FROM: IMAGE: (121112,DCM,"Source of Measurement")  = (1.2.840.10008.5.1.4.1.1.2,1.3.6.1.4.1.14519.5.2.1.8421.4008.767475413701844560980492237110)
```

Example 1 – JSON of the content tree (only)


```
"ImagingMeasurementReport": [
  {
    "_tmr": "DCMR",
    "_tid": "1500"
  },
  [
    {
      "ImagingMeasurements": [
        [
          {
            "MeasurementGroup": [
              [
                {
                  "Length": [
                    {
                      "_units": "mm"
                    },
                    "97.08595644",
                    [
                      {
                        "Path": [
                          {
                            "_gtype": "POLYLINE",
                            "_coord2d": [
                              186.4132537841797,
                              274.5900573730469,
                              89.10497283935547,
                              374.7270812988281
                            ]
                          },
                          [
                            {
                              "SourceOfMeasurement": [
                                {
                                  "_class": "CTImageStorage",
                                  "_instance": "1.3.6.1.4.1.14519.5.2.1.8421.4008.767475413701844560980492237110"
                                }
                              ]
                            }
```

Example 1 – JSON of result only (no coords)


```
"ImagingMeasurementReport": [
  {
    "_tmr": "DCMR",
    "_tid": "1500"
  },
  [
    {
      "ImagingMeasurements": [
        [
          {
            "MeasurementGroup": [
              [
                {
                  "Length": [
                    {
                      "_units": "mm"
                    },
                    "97.08595644"
                  ]
                }
              ]
            ]
          }
        ]
      ]
    }
  ]
```

Example 1 –Business Names file (partial)


```
{"ImagingMeasurementReport": {
    "_cv": "126000",
    "_csd": "DCM",
    "_cm": "Imaging Measurement Report",
    "_vt": ["CONTAINER"]
}},
{"Liver": {
    "_cv": "10200004",
    "_csd": "SCT",
    "_cm": "Liver"
}},
{"PersonObserverName": {
    "_cv": "121008",
    "_csd": "DCM",
    "_cm": "Person Observer Name",
    "_vt": ["PNAME"],
    "_rel": ["HAS OBS CONTEXT"]
}},
```

Keywords for UIDs


```
"SOPClassUID": {
  "Value": [
    "1.2.840.10008.5.1.4.1.1.88.22"
  ]
},
```



```
"SOPClassUID": {
  "Value": [
    "EnhancedSRStorageSOPClass"
  ]
},
```

Out of Scope (for this development cycle)

- A DICOMweb API to transform JSON SR to/from the standard binary DICOM SR persistent form, though experimental media types are defined (in a WADO-RS or STOW-RS “application/dicom+json” like manner, e.g., “application/x-dicom-sr+json”)
- A DICOMweb API to access, create or modify the DICOM SR content tree abstraction (cf. the existing RetrieveMetadata individual DICOM attribute level access)
- A DICOMweb API to create and manage individual (or sets of) annotations separately from the storage/retrieval of entire DICOM SR object
- A DICOMweb API to perform/manage the various steps of the authoring pipeline that adds lesion management, image references and descriptions, and patient/study/series/workflow composite context