Digital Imaging and Communications in Medicine (DICOM)

Sup 219 - JSON Representation of DICOM Structured Reports

DICOM Standards Committee - Working Group 23 - Artificial Intelligence/Application Hosting

1300 N. 17th Street Suite 1752
Rosslyn
VA
22209
USA

Status: Public Comment
Publication date 2019/11/07
This supplement is prepared pursuant to work item: 2019-04-A
Copyright © 2019 NEMA
# Table of Contents

Document History ........................................................................................................ 6
To Do After Public Comment ....................................................................................... 7
Open Issues .................................................................................................................. 8
Closed Issues ................................................................................................................ 8
Scope and Field of Application ..................................................................................... 15
XXX. Transformation of JSON Representation of Structured Reports (Informative) .... 18
XXX.1. Background ....................................................................................................... 18
XXX.2. Example of Successive Refinement ................................................................. 18
PS3.23 .......................................................................................................................... 27
Notice and Disclaimer ................................................................................................. 28
Foreword ....................................................................................................................... 29
1. Scope and Field of Application ................................................................................ 30
2. Normative and Informative References .................................................................. 31
3. Definitions ................................................................................................................ 32
3.1. Codes and Controlled Terminology Definitions: ................................................. 32
3.1. Representation Conversion Definitions: ............................................................... 32
4. Symbols and Abbreviations .................................................................................... 33
5. Conventions ............................................................................................................. 34
A. XML Encoding ......................................................................................................... 62
A.1. Introduction to XML ............................................................................................. 61
A.2. XML Encoding of DICOM Instances ................................................................. 35
A.2.1. DICOM XML Encoding in Native Format ....................................................... 35
A.2.1.1. Usage ............................................................................................................. 35
A.2.1.2. Identification ................................................................................................. 35
A.2.1.3. Support ......................................................................................................... 35
A.2.1.4. Information Model ....................................................................................... 35
A.2.1.5. Description ................................................................................................. 35
A.2.1.6. Schema ........................................................................................................ 35
A.2.1.7. Examples ..................................................................................................... 35
B. JSON Encoding ........................................................................................................ 36
B.1. Introduction to JavaScript Object Notation (JSON) ............................................. 36
B.2. JSON Encoding of DICOM Instances and Messages ......................................... 36
B.2.1. Introduction ..................................................................................................... 36
B.2.2. DICOM JSON Encoding .................................................................................. 36
B.2.3. Transformation to and from other DICOM Encodings ................................... 36
B.2.4. DICOM JSON Encoding Example ................................................................. 36
B.3. JSON Encoding of Structured Reports ................................................................. 36
B.3.1. Introduction ..................................................................................................... 36
B.3.2. DICOM JSON Structured Report Encoding .................................................... 36
B.3.2.1. Attribute Encoding .................................................................................... 37
B.3.2.2. Structured Report Content Tree Encoding ............................................... 39
B.3.2.2.1. Content Item Encoding ............................................................................ 39
B.3.2.2.2. Nested Content Encoding ...................................................................... 39
B.3.2.2.3. Content Item Annotations .................................................................... 40
B.3.2.2.4. Encoding of By-Reference Relationships .................................................. 41
B.3.2.2.5. Encoding of Content Items of Specific Value Type .................................. 41
B.3.2.2.5.1. Encoding of Content Items Without a Value ...................................... 42
B.3.2.2.5.2. Encoding of Content Items with a Single Value .................................. 42
B.3.2.2.5.3. Encoding of Numeric Content Items ...................................................... 43
B.3.2.2.5.4. Encoding of Content Items That Reference Storage SOP Instances .... 43
B.3.2.2.5.5. Encoding of Coordinate Content Items ................................................. 45
B.3.2.2.6. Encoding of Business Names File ............................................................ 46
B.3.3. DICOM JSON Structured Report Encoding Examples .................................... 48
List of Figures

2 XXXX.1-1. Example of Successive Refinement of JSON Payload to Complete SR .......................................................... 18
3 XXXX.1-2. Example of Single Linear Measurement to Encode in SR ............................................................................ 18
List of Tables

1. TID 1500. Measurement Report ........................................................................................................ 62
# Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Date</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>2019/07/09</td>
<td>First draft for review by WG 23</td>
</tr>
<tr>
<td>02</td>
<td>2019/07/19</td>
<td>Changes after review by WG 23 and other feedback received</td>
</tr>
<tr>
<td>03</td>
<td>2019/07/22</td>
<td>Changes after feedback received (KH, AF)</td>
</tr>
<tr>
<td>04</td>
<td>2019/09/08</td>
<td>For WG 6 first read</td>
</tr>
<tr>
<td>05</td>
<td>2019/09/09</td>
<td>After WG 6 first read</td>
</tr>
<tr>
<td>06</td>
<td>2019/11/02</td>
<td>Assigned supplement number; new part is 23 not 22 (which was assigned to RTV); add Business Names File description; add PS3.17 example of pipeline with successive refinement of JSON; collapse value arrays into single strings in content tree (not yet data elements).</td>
</tr>
<tr>
<td>07</td>
<td>2019/11/03</td>
<td>More on open issues, including JSON-LD, positional versus parametric representation of coordinates etc., use of keywords and business names in place of UIDs, number of business names files and whether they should be explicitly referenced; more ambiguity resolution rules; move informative annex to front of document.</td>
</tr>
<tr>
<td>08</td>
<td>2019/11/07</td>
<td>WG 6 review 2019/11/06: update to do items, open and closed issues; correct typos; improve scope and forward text, collapse value arrays into single strings in data elements as well as content tree, add example of empty (zero length) value for data element; add illustration with CT image and measurement for example.</td>
</tr>
<tr>
<td>09</td>
<td>2019/11/07</td>
<td>Public Comment.</td>
</tr>
</tbody>
</table>
# To Do After Public Comment

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fill in hyperlinks to other parts, and especially add them to PS3.3 descriptions of each Value Type.</td>
</tr>
<tr>
<td>2</td>
<td>Example and test tool round trip - add DICOM data elements to business names file as described (including private data elements and creators)</td>
</tr>
<tr>
<td>3</td>
<td>Text, examples and test tool round trip - add NUM Content Item Annotations</td>
</tr>
<tr>
<td>4</td>
<td>Text, examples and test tool round trip - add IMAGE Content Items with frame numbers (esp. array of multiple) and segment numbers, etc.</td>
</tr>
<tr>
<td>5</td>
<td>Text, examples and test tool round trip - add SCOORD other Attributes as Annotations or positionally, specifically PixelOriginInterpretation (for WSI) and FiducialUID</td>
</tr>
<tr>
<td>6</td>
<td>Text, examples and test tool round trip - add SCOORD3D Content Items</td>
</tr>
<tr>
<td>7</td>
<td>Text, examples and test tool round trip - add WAVEFORM and TCOORD Content Items and distinguish ReferencedSamplePositions, ReferencedTimeOffsets or Referenced DateTime</td>
</tr>
<tr>
<td>8</td>
<td>Test parsing of TEXT/CODE ambiguity for Comment, or Equivalent Meaning in TID 300 (CP 1929)</td>
</tr>
<tr>
<td>9</td>
<td>Need to exhaustively test to make sure ambiguity resolution rules are defined and robust, else require annotation of @vt or @rel inline in JSON file</td>
</tr>
<tr>
<td>10</td>
<td>Check Number versus String (DS NUM versus FD coordinates for consistency with PS3.18 Annex F; need to maintain alignment with work in progress CP 1861, CP 1889.</td>
</tr>
<tr>
<td>11</td>
<td>Add annotations for extended attributes and long or URL codes and alternative codes.</td>
</tr>
<tr>
<td>12</td>
<td>Add SCOORD annotations for other attributes like PixelOriginInterpretation (for WSI) and FiducialUID.</td>
</tr>
<tr>
<td>13</td>
<td>Complete description of SCOORD3D and associated annotations.</td>
</tr>
<tr>
<td>14</td>
<td>Complete description of TCOORD and associated annotations related to TemporalRangeType, and positions of ReferencedSamplePositions, ReferencedTimeOffsets or Referenced DateTime in array.</td>
</tr>
<tr>
<td>15</td>
<td>Add annotations for all content items to support ObservationDateTime and ObservationUID (PS3.3 Table C.17-6)</td>
</tr>
<tr>
<td>16</td>
<td>Add media type for transformed JSON content ? one for snippet versus entire instance? Should await web service extensions in future work item? Although a media type, e.g., application/dicom-sr+json or similar may be defined, arguably just application/json should always be used, but then we have already established the use of application/dicom+json and application/dicom+xml for the PS3.18 metadata, and registered these successfully with IANA; see also HL7 discussion <a href="https://wiki.hl7.org/index.php?title=Media-types_for_various_message_formatsand">https://wiki.hl7.org/index.php?title=Media-types_for_various_message_formatsand</a> IETF XML discussion <a href="http://www.rfc-editor.org/rfc/rfc3023.txt">http://www.rfc-editor.org/rfc/rfc3023.txt</a>.</td>
</tr>
</tbody>
</table>
Open Issues
Comments are sought on the choice of positional versus named parameters.

The terseness of positional rather than named object descriptions of various content items, especially for coordinates and image references, has been emphasized over clarity and human readability (otherwise we would use XML). The current proposal for an anonymous SCOORD content item with a subordinate IMAGE reference uses positional parameters like this:

```json
"": [
  "POLYLINE",
  [172.83535766601562, 270.0640869140625, 133.7988916015625, 343.0453186035156]
],
[
  
],
[
  
]
```

Various alternative suggestions have been received, something like this perhaps, where the annotation symbol "@" is used to distinguish annotations from business names, the purposes of various parameters are explicitly indicated, and the value types of the anonymous content items are explicitly indicated with annotations, in the absence of a business for the concept name to indicate (and look up) what the value type should be:

```json
"": {
  "@vtype": "SCOORD",
  "@gtype": "POLYLINE",
  "@coords": [
    172.83535766601562,
    270.0640869140625,
    133.7988916015625,
    343.0453186035156
  ],
  [
    
  ],
  [
    
  ]
}
```

By way of comparison, GeoJSON (https://geojson.org/) uses something like the following (without the image references, since GeoJSON uses longitude and latitude in the WGS 84 geographic coordinate reference system):

```json
{
  "...,
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  }
}
```
It may also be useful to compare with the GraphQL [https://graphql.org/](https://graphql.org/) approach.

Some people have strong opinions about typing, e.g.: [https://tech.signavio.com/2017/json-type-information](https://tech.signavio.com/2017/json-type-information).

Concern has been expressed about "anonymous" content items (those with no concept name to use as a business name, therefore sent as ""), especially when there is no explicit Value Type (e.g., IMAGE), being potentially confusing and error-prone. However, the standard for the underlying SR infrastructure allows for "anonymous" content items (no concept name) and the templates (like TID 1500 sub-templates) allows for them too, so they are something the JSON representation has to support. The proposed syntax is trivial, an empty quoted string where the business name would go. Is this sufficiently clear, once one accepts that "anonymous" content items have to be supported?

Is there a need to allow annotations with explicit Value Type and/or Relationship Type for cases where either there is no Business Name for the Concept Name (anonymous Content items, often used for IMAGE), or there is ambiguity (e.g., same Business Name used with two different Value Type and/or Relationship Type)? In the case of anonymous SCOORD content items, SELECTED FROM and INFERRED FROM relationships with child images and parent TEXT, CODE or NUM content items can be assumed due to IOD-specified relationship constraints. Ideally, developers would not need to be bothered by relationships.

Should we insist on different business names for the same code used as the concept name for different value or relationship type scenarios?

Some would prefer more full names rather than abbreviations for content item or business name annotations. We currently think that terseness is a priority over readability, and that developers will rapidly learn the abbreviated constructs, so use, for example, "@cv" instead of "@CodeValue", or "@ref" instead of "@ReferencedContentItemIdentifier."

Should there be standard default business names (e.g., defined in a PS3.6 keyword-like manner for every code used in PS3.16)?
Should there be an "include" mechanism for business name files?

Currently the business names file location and identity is implicit (out of scope of the standard to define how they are associated, found, managed). By comparison, JSON-LD (https://www.w3.org/TR/json-ld/#the-context) allows an explicit reference of the form:

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  ...
}
```

and we could do something similar:

```
[
  {
    "@businessnames": "http://acme.org/businessnames/measurement_businessnames.json",
    "SOPClassUID": "1.2.840.10008.5.1.4.1.1.88.22",
    ...
  }
]
```

or

```
[
  {
    "@businessnames": "file://acmeproduct/businessnames/measurement_businessnames.json",
    "SOPClassUID": "1.2.840.10008.5.1.4.1.1.88.22",
    ...
  }
]
```

However, such absolute or relative links may go stale, and are not be consistent with the general expectation that DICOM objects are portable and do not depend on any particular location and are identified by UIDs, not names or URLs. Association of the files is thought to be an architectural issue that should be deferred until DICOMweb APIs specific to JSON SR handling are defined, but input is sought on the pros and cons of defining an include mechanism.

Do we need explicit support for JSON-LD (https://en.wikipedia.org/wiki/JSON-LD) for business names, or allow JSON-LD annotations to be present and be ignored in the encoded JSON file and/or the business names JSON file? There is currently no prohibition on there being a separate JSON-LD context file present that describes the business names used. The current draft reserves the use of any object name beginning with "@" in the results JSON file or business names file however.


JSON-LD information can be provided in a separate JSON document than the JSON document whose objects it is describing, a so-called JSON-LD Context File, which may or may not be referenced from the affected file (https://www.w3.org/TR/json-ld/#the-context). Using the separate JSON-LD content file approach with DICOM JSON files would allow the business names to also be described by JSON-LD links, in addition to the conventional coding scheme based codes (or URL codes) in the proposed JSON business name file. Feedback is sought on the value of explicitly documenting JSON-LD support as opposed to just not preventing its use.

Should business name definitions be allowed in same file as the content? Currently it is proposed to enforce that they are separate, to avoid two ways to do the same thing (since we definitely want to support separate files for commonality and re-use), and to avoid the need for yet another complication of the main content syntax.

Should more than one business name definitions file be permitted (e.g., to have a standard list of data elements, a separate standard list of codes, and a set of local or instance specific customizations)? Currently the text implies that there is only one, but in the absence of a service definition, restriction to a single file or a requirement to support more than one are probably out of scope.
Should business names be used for header Code Sequence Items in Code Sequence Attributes in the top level Data Set. Currently they are encoded in the traditional manner, i.e., as individual DICOM Attributes, (a) to align the DICOM Attribute header as closely with PS3.18 Annex F as possible, and (b) very few, if any, Code Sequence Items are used in the headers of DICOM Structured Reporting SOP Classes. Should business names be used instead?

Do we need support for business name prefixes and/or name spaces? Business names may be any string, no standard prefixes are defined (perhaps with the exception of forbidding some keywords for annotations beginning with `@`), there is no standard namespace mechanism in JSON and nor is it apparent that we need one. "Clashes" are not an issue because the scope of uniqueness is the pair of encoded JSON file and business name JSON file, and so the same business name can be used for another pair with a different meaning.

Suggestions are sought on how to use existing standards to formalize the syntax that is permitted, such as JSON Schema. Is it possible to write a JSON Schema that defines only the structural rules without being dependent on the data element keywords or coded concept business names. Preliminary experiments at defining a JSON Schema for the syntax and rules, but found this hard because since the objects are defined by template-specific structure and instance-specific business names. E.g., one can try feeding the sample into https://www.liquid-technologies.com/online-json-to-schema-converter or https://jsonschema.net/. A JSON schema for a very specific use of a specific template using standardized business names (e.g., for TID 1500 used in a particular way) would be straightforward to define. Is there an alternative to JSON Schema for formal representation of the rules?

TID 1500 requires a Language, Procedure Reported, an empty section (CONTAINER) for the Image Library, and both Tracking Identifier and Unique Identifier, all of which complicate the "simplest" example - should TID 1500 be modified to relax some of these requirements? E.g., it would be harmless to omit an empty Image Library CONTAINER), and Procedure Reported may not be known and will often be a dummy value based on Modality only, and the Language is irrelevant for measurements (though not for code meanings of coded concepts, but could be assumed or just left unstated). The Tracking Identifier and Unique Identifier are probably important to retain though.

This is proposed in the current draft.
Should we use business name or standard defined keywords in place of UIDs such as SOP Class UIDs?

It would be simpler for developers to use "CTImageStorageSOPClass" rather than "1.2.840.10008.5.1.4.1.1.2", and "EnhancedSRStorageSOPClass" rather than "1.2.840.10008.5.1.4.1.1.88.22". To implement this, the generator/parser would have sufficient context that it knows that it is expecting a UID and could perform the necessary substitution/lookup.

The keywords could easily be standardized for all UIDs defined in PS3.6, in the same manner as they were standardized for the data elements by CP 850.

Business names could be used for private UIDs, or those standard UIDs the implementer is concerned might be too new to be recognized by recipients, or to provide localized language synonyms.

Feedback is sought on whether or not this optimization is worth the effort.

E.g., instead of:

"SOPClassUID": "1.2.840.10008.5.1.4.1.1.88.22"

we could have:

"SOPClassUID": "EnhancedSRStorageSOPClass"

and we could have entries in the business names file like:

"AcmeRawDataStorageSOPClass": {
   "@uid": "1.2.3.4.5.97456145",
   "@uidtype": "SOPClass"
   "@serviceclass": "Storage"
}

Getting really carried away, one could allow business names for all UIDs, including Study, Series or SOP Instance, to allow business names like "currentimage" to be used in place of references to the SOP Instance UID, and allow the business names file (or perhaps a separate business names file than the one used for codes and data elements) to be used to make the association with the actual image. This may be too creative and a step too far (if not potentially unsafe). Conversely, prohibiting this requires treating the UID substitution mechanism as a special case rather than applicable to all UIDs, which may require more effort rather than less for the generator/parser.

Do we want to allow ""StudyDate": null" and/or ""StudyDate": "" instead of ""StudyDate": {} for zero length Attributes (the empty object currently proposed being a consequence of following the current PS3.18 Annex F representation but eliding the vr string and value arrays)?

For private data elements, should creator element insertion be automatic/transparent or explicit/manual (creator element needs to be included)? Currently it is explicit/manual, to be consistent with current PS3.18 Annex F representation.

Should the PNAME Value Type JSON encoding be decomposed into its components and component groups, as is done for the DICOM Attribute PN VR per the current PS3.18 Annex F description? Probably. Right now it is just a simple single JSON String. Alternatively, should we stop using the PS3.18 decomposition for the top level Attributes, as some have suggested? The reason this was started in the WG 23 PS3.19 XML encoding was to avoid parsers needing to be aware of the DICOM-specific (HL7-like) use of caret and equal symbols as delimiters and to leverage the mechanisms inherently present in XML (or JSON).

Do we need a generic solution for including unrecognized private Attributes attached to Content Items in the Content Item Annotations description? Currently these would be omitted, since the Content Tree is not handled the same was as top level data set data elements. Annotations could be used for this but it would be ugly.

Should the top level DICOM data elements be encoded as a direct child of the top-level array, along with the Content Tree? The current approach follows the example in PS3.18 Annex F.4, which shows multiple query results, each of which is an object in the top level array. While this Annex F complication could be elided, it would reduce the reusability of parsers/generators designed to handle both.
|   | Does anything need to be said about the ability to reference into the JSON content, e.g., with a JSON Pointer (http://tools.ietf.org/html/rfc6901)? Probably not, since any named JSON Object within a JSON representation of an SR can be identified by its business name and that business name used in the JSON Pointer syntax. Further, as yet, no use cases have been identified that require external references into content items with a JSON SR, as opposed to the entire SR. Note also that any content item in a DICOM SR can be decorated with an ObservationUID, which could be exposed via an annotation in the JSON file, and used as a reference target. |
Closed Issues

1. A new Part is needed, since there is no good home for this transformation. All the alternative representations be gathered in this new Part, specifically PS3.19 A.1 and PS3.18 Annex F. The PS3.19 A.2 Abstract Multi-Dimensional Image Model has not been moved. The previously named "Model" is renamed as "Encoding". Confirm. Consider "Representation" or "Format" rather than "Encoding".

2. It is necessary to include the metadata associated with the SR content. The header is required eventually to make a valid DICOM object that can be stored in the PACS. The example shows a multi-step process: first generate the SR content tree, then add the header (a separate tool, not the AI result creator, can do this, given context and the original DICOM image headers). In the absence of a "platform" (which we are not yet defining), this work must occur out of band. The preamble to the document (and work item) describes such a platform as might be based on DICOMweb as out of scope for now, but you could imagine a service that added the JSON content tree (only) to an existing DICOM study and that filled in the header. An informative annex is added that shows a multi-step pipeline that successively refines the content.

3. Use line numbering in JSON examples in the PDF, even though it prohibits select/copy/paste for experimentation. The tooling doesn't permit line numbers to be turned off just for the examples, and they are need for reference by commentators. Suggest using the XML DocBook source if you want to copy/paste. The final standard will have no line numbers.

4. A general compact JSON model has not been used in favor of using a specific SR JSON model. One could remove "vr" and flat "Value" properties, encoding Person Names according PS3.5, and use JSON Pointers in URI fragments (https://tools.ietf.org/html/rfc6901#section-6) to include content from other (possibly remote) documents. This is not within the scope of the work item, which is specifically about simplifying the representation of the SR content tree, but is a subject that may be taken up by WG 27.

5. Keywords are used in place of hex tags, even though keywords do not work for private tags (without business names for them), repeating attributes (e.g. Overlays), and require a constant update of attributes after each new release of DICOM standard. The use of keywords rather than hex tags in the SR representation seems to be a popular idea, and using keywords makes things less awful for AI developers. SRs rarely, if ever, contain private data elements, very rarely have new data elements, and do not contain repeating groups like overlays. The supplement addresses using business names for private data elements and new data elements in the unlikely event that they are needed.

6. There is no need for the value (of a data element or an SR content item) to always being a JSON Array, when it is a leaf node and the value is a single JSON String representing a text value or a coded value Business Name. A single value is allowed instead, for both top level data set Attributes and for SR Content Items. This very common simplification is important enough to deviate from PS3.18 Annex F.

7. Regarding the IMAGE positional argument: in the situations where multiple items can be present, null will be used for those that are missing. "Trailing unused values may be elided but intervening values are required to be null if there is no value, in order to preserve the positional order."

8. In the example, there are some Values entries missing in header attributes. This is intentional, since that is how PS3.18 Annex F (existing JSON) encodes Type 2 (empty) Attributes.

9. In the example, in the Image Library, there are a lot of nested brackets and parentheses. That's because of the level of nesting of child content items and how they are encoded as objects in arrays and the additional level of arrays required to handle sibling content items with non-unique concept names.

10. The result content file is documented before the business names file, even though the former depends on the latter, since the design and structure of the result content file is of more interest to the reader and the business names file is more administrative and routine.
Scope and Field of Application

This Supplement describes a JSON representation of DICOM Structured Reports, similar to the PS3.18 JSON representation, to allow AI developers to encode image-derived results. Patterns are defined for transformation of measurement and annotation information for use-cases related to the reporting of artificial intelligence (AI), machine learning (ML) and quantitative imaging (QI) results. The approach is applicable not only to export of AI/ML results but also to encoding of truth data for AI/ML training, testing and validation.

JSON has emerged as the preferred representation for results from machine learning algorithms amongst developers who are not familiar with the DICOM Standard. Such results typically lack the composite context information required in a managed clinical environment (such as patient and study identity information), as well as references to the DICOM images used as input, needed to store, distribute and render the results on an image viewing system. DICOM Structured Reports (SR) are the most common form of semantically meaningful annotation created and distributed in an interoperable manner for clinical use. Accordingly, this supplement describes a mapping between a JSON representation of the measurement and annotation result payload expected from an AI system, and the traditional binary DICOM SR encoding of the same information.

DICOM PS3.16 defines templates for different applications of SR. The TID 1500 Measurement Report template describes a generic pattern that is suitable for encoding AI results as well as other quantitative and qualitative (categorical or descriptive) results. The JSON representation in this Supplement is exemplified using TID 1500, but the conversion supports full semantic fidelity round-trip encoding of any DICOM SR instance, regardless of the template.

Using an appropriate tool, a complete and compliant binary SR can be created automatically from the JSON, with the subtleties of DICOM encoding hidden from the user. The JSON representation may be useful beyond the primary AI application that motivated the work. It is not expected that the JSON representation will be used as the persistent form, but rather that the existing DICOM binary object storage infrastructure will be used.

A multi-step process for transformation is envisaged. First, the result payload itself may be encoded in JSON; this is limited to the minimal necessary information to describe the result itself, for simplicity and ease of use by AI/ML algorithm developers. Then, this result JSON is merged with the necessary JSON representation of the composite context and other mandatory, or relevant optional, SR content (such as UIDs, image libraries, hierarchical identification and report status management information), which, when transformed, would result in a valid SR IOD with a template-compliant content. Finally, the JSON is transformed into the traditional binary DICOM SR representation for transport, storage and management in an interoperable form. An Informative Annex describing such a "successive refinement" approach is included.

It is anticipated that future Supplements will extend the DICOMweb services to support transformation of JSON DICOM SR into binary DICOM SR, and to retrieve transformed content, e.g., by leveraging the existing STOW-RS and WADO-RS mechanisms. The scope of this Supplement is limited to describing the representation and the transformation.

The JSON representation leverages the "business name" concept from HL7 Green CDA, such that short meaningful strings can be used in the JSON for coded tuples for concept names and values, as well as for DICOM Attributes in the top level Data Set. The business names are defined in a separate, potentially reusable, JSON file, which may be user- or organization-supplied or automatically generated.

DICOM SR traditionally makes very extensive use of coded terminology to maximize semantic interoperability and to avoid reinventing existing content. However, choosing and encoding codes can be burdensome to developers. Similarly, the use of DICOM Data Element tags rather than keywords can be similarly confusing. Accordingly, "business names" are used as a substitute for the more arcane codes and tags that are normally used. For example, "StudyDate" may be used in the JSON representation in place of (0008,0020), and "Length" may be used as opposed to (410668003, SCT, "Length"). The business names to be used can either be supplied by the creator of the JSON representation, some other organization or authority, or selected from a standard list of keywords from the DICOM Data Dictionary (PS3.6) or business names for concepts provided in the DICOM Content Mapping Resource (DMR) (PS3.16). The business names are not qualified by any prefix or namespace in the interests of terseness and simplicity. The scope of uniqueness of the business names only has to encompass the encoded JSON file and its accompanying business names JSON file. That said, business names may be as complex a string as the user cares to create, and any current or future convention for pseudo-name space mechanisms can be utilized if desired.

The choice of JSON representation leverages the existing PS3.18 Annex F JSON representation of ordinary DICOM objects at the data element level, though with the use of keywords in place of data element tags for clarity and simplicity. The DICOM SR content is divided into two components, the data elements representing the SR content tree, which are encoded as if each node (content item) of the content tree were a distinct entity, and the remainder of the DICOM data elements that are normally encoded in the top level DICOM data set. When parsing the JSON representation, all DICOM keywords that are recognized and that are not part of the
SR content tree are extracted, then all remaining content is examined for matches with the defined business names. To the extent possible, the relationships and value types that are defined for the DICOM binary SR representation are elided from the JSON representation, and either defined with the business names, or inferred from context. For example, whether a coded concept has a CONTAINS or HAS CONCEPT MOD relationship with its parent CONTAINER value type is not something an AI/ML developer is likely to be concerned about, and requires a level of DICOM expertise that they are unlikely to have. Accordingly, a coded concept that is always used as a concept modifier can have this declared in the business name descriptor rather than repeating the information in every use in the JSON payload. Similarly, some concepts always have a predictable value type (e.g., of CODE or TEXT or NUM) that can be declared in the business name file. Occasionally, SR content items have no required concept name (e.g., IMAGE references within SCOORD) but such patterns can be detected and inferred. When ambiguity is possible, then the JSON representation can be made explicit to resolve it (e.g., to distinguish TEXT from CODE value types when either may be used and there is ambiguity as to whether the value should be used literally or looked up as a business name for the code value).

In the interest of simplicity, the JSON representation chosen is largely positional rather than using named parameters; e.g., the payload of a content item that is a TEXT or CODE value type is an array of strings, which in the degenerate form is a single value, wherein the string is used as a TEXT value or a CODE business name depending on what the concept name is recognized to be, rather than explicitly naming the type of value (i.e., “Kidney” rather than “{@code”: “Kidney”). Similarly, more complex content item value types like NUM or SCOORD or IMAGE, for which there are multiple values with a specific purpose are defined positionally as well, e.g., the position in the array determines whether the business name is for the units, or the UID is for the SOP Class or SOP Instance, or whether the value is the graphic type or an array of coordinates (and for the latter, which are X and Y), etc. This places greater demands on the parser/converter as well as the documentation, but provides simplicity (if not always clarity) for the user. Alternatives such as fully qualifying every parameter with its type and purpose produced an undesirably cluttered representation, especially for the simple use cases.

At this time, no similar standard XML representation is defined, thought the concepts are equally applicable, theoretically. The consensus in the AI/ML community at this time seems to be to focus on JSON. Several toolkits have their own XML schemas and representations for DICOM SR, but there has been no significant effort to harmonize or standardize these, and the outstanding work item to do so (2012-11B) has been withdrawn after many years of inaction.

This Supplement defines a new Part, PS3.23, to contain the alternative representation, and includes the existing Attribute level XML and JSON encodings factored out of PS3.19 and PS3.18.
XXX.1 Background

A reliable and interoperable interchange framework for the communication of results requires not only that a means of encoding the result payload be defined, but that the result be accompanied by the necessary metadata to allow its management in a patient-related and workflow-related context, even when the result is detached from the system in which it is managed. Just as for DICOM images, this result metadata is defined according to the DICOM Information Model, and it is encoded in the corresponding Composite Instances that are the persistent, interchangeable representation of DICOM Structured Reports.

However, it is recognized that result creation and rendering systems may be modular in their design, such that one component of a system may not be aware of, or need to be aware of, certain types of information.

As a case in point, a machine-learning-based algorithm that generates a numerical result, such as probability of malignancy, may need no knowledge of a patient’s identifying or descriptive metadata. Conversely, a result management system, to which the exact nature of the result payload is essentially opaque, needs the identifying metadata but may be agnostic to result payload structure and content that may be supplied by different algorithms.

Accordingly, the JSON representation of Structured Reports has been designed to allow for such modularization and division of responsibility for content generation.

XXX.2 Example of Successive Refinement

This Section describes an example of successive refinement of a relatively simple numerical payload with image-related, lesion, and study and patient-related metadata.

Figure XXXX.1-1 illustrates an example of a pipeline that might used to take algorithm-generated result content in its most minimal form through several successive stages, adding the necessary metadata to generate a complete JSON representation of a valid DICOM Structured Report, which is then converted into its traditional binary representation and sent to an ordinary DICOM Storage SCP.

Note

The components in this pipeline need not be physically co-located or on-premise. E.g. the algorithm could execute on a cloud-based computing resource using anonymized data, and the context information be managed and added on-premise when the results were returned.

The AI Algorithm in this example may produce a very simple numeric result, such as the single linear dimension of a tumor illustrated in Figure XXXX.1-2, but still one that is linked to the image coordinates from which it is derived, as follows:
"MeasurementGroup": [
    {
        "Length": [
            "66.43856134", 
            "mm",
        ]
        "": [
            "POLYLINE",
            [ 
                172.83535766601562, 
                270.0640869140625, 
                133.79888916015625, 
                343.0453186035156
            ],
            [ 
                "1.2.840.10008.5.1.4.1.1.2",
                "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842228818573226516023762"
            ],
        ]
    },

    {
        "": [
            "1.2.840.10008.5.1.4.1.1.2",
            "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842228818573226516023762"
        ]
    }
]

Note

1. In reality, many machine learning and quantitative algorithms operate on image pixel data that has been extracted from its DICOM form. Hence the algorithm software may be unaware of the SOP Instance UID and SOP Class UID of the image. It is expected that at the very least such algorithms will be wrapped by a management system that maintains the correspondence between the DICOM UIDs and the images used by the algorithm.

2. Such algorithms may make use of coordinate representations that do not exactly match the DICOM sub-pixel resolution 2D or patient-relative, volume-relative or slide-relative 3D coordinate systems. It is expected that such algorithms will be wrapped by a management system that transforms the coordinates into the standard form as necessary.

3. In this example, the AI algorithm is producing only content encoded according to a nested content template, which is intended to be later embedded in a more complete root template such as TID 1500. As such, the root content item that it produces starts at the MeasurementGroup (ROI) level, not the ImagingMeasurementReport level.

4. There is no lesion tracking information provided, since this particular hypothetical algorithm is assumed to be unaware of longitudinal temporal relationships.

Next, the Lesion Manager adds longitudinal lesion tracking information and a finding site, since it has access to out-of-band information related to other lesions measured on different time points:

"ImagingMeasurements": [
    {
        "MeasurementGroup": [

        ]
    }
Next, a DICOM-Image Aware System wraps the nested MeasurementGroup level content into a root-template, such as an ImagingMeasurementReport, and adds contextual information that includes:

- identification of the template used
- language information
- Image Library entries corresponding to the referenced images
- information about the Procedure Reported
- Person Observer Context

```
  "ImagingMeasurementReport": [
    {
      @tmr": "DCMR",
      "@tid": "1500"
    },
```
Finally, a Patient-Study Aware System takes the Structured Report Content Tree from the previous stage, and adds the necessary top-level DICOM Data Set Attributes to produce a valid DICOM SOP Instance compliant with the Enhanced SR Storage SOP Class, initially in its JSON Representation.

This step includes adding such contextual information as:

- Patient identifying and descriptive metadata
- Study identifying and descriptive metadata, including an appropriate StudyInstanceUID, such as that extracted from the referenced image(s)
- Equipment identifying and descriptive metadata
- new Series identifying and descriptive metadata, including an appropriate new SeriesInstanceUID
- new Instance identifying and descriptive metadata, including an appropriate new SOPInstanceUID and an appropriate SR Storage SOP Class UID
- additional DICOM Attributes and Values necessary to conform to the SR Storage SOP Class, including:
  - additional Person Observer Context information in the AuthorObserverSequence
  - report status management information, such as the CompletionFlag and VerificationFlag
  - the evidence sequence(s) necessary to provide a full hierarchical UID-based route to the reference image content, to support a hierarchical rather than relational query (DIMSE C-FIND or QIDO-RS), such as the CurrentRequestedProcedureEvidenceSequence
["
  "SOPClassUID": "1.2.840.10008.5.1.4.1.1.88.22",
  "SOPInstanceUID": "1.3.6.1.4.1.5962.1.1.0.0.0.1572701186.46156.1",
  "StudyDate": "19870620",
  "SeriesDate": {},
  "ContentDate": "20171126",
  "StudyTime": "135823",
  "ContentTime": "224217",
  "AccessionNumber": {},
  "Modality": "SR",
  "Manufacturer": "PixelMed",
  "InstitutionName": {},
  "ReferringPhysicianName": {},
  "StationName": "NONE",
  "StudyDescription": "Renal",
  "SeriesDescription": "Crowds Cure Cancer Annotation as Measurement Report",
  "ManufacturerModelName": "XSLT from annotations_expanded.csv",
  "ReferencedPerformedProcedureStepSequence": {},
  "PatientName": {
    "Value": [
      {
        "Alphabetic": "TCGA-BP-4343"
      }
    ],
    "PatientID": "TCGA-BP-4343",
    "PatientBirthDate": {},
    "PatientSex": {},
    "DeviceSerialNumber": "9723613413261",
    "SoftwareVersions": "0.1",
    "StudyInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.100545948082587796019777812429",
    "SeriesInstanceUID": "1.3.6.1.4.1.5962.1.3.0.0.1572701186.46156.1",
    "StudyID": {},
    "SeriesNumber": "4578",
    "InstanceNumber": "1",
    "AuthorObserverSequence": {
      "Value": [
        {
          "InstitutionName": {},
          "InstitutionCodeSequence": {},
          "PersonIdentificationCodeSequence": {},
          "ObserverType": "PSN",
          "PersonName": {
            "Value": [
              {
                "Alphabetic": "accomplished_peafowl"
              }
            ]
          }
        }
      ],
      "PerformedProcedureCodeSequence": {},
      "CurrentRequestedProcedureEvidenceSequence": {
        "Value": [
          {
            "ReferencedSeriesSequence": {
              "Value": [
            ]
          }
        }
      }
    }
  }
]
ReferencedSOPSequence: {
  "Value": [
    {
      "ReferencedSOPClassUID": "1.2.840.10008.5.1.4.1.1.2",
      "ReferencedSOPInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"
    }
  ],
  "SeriesInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.1780604360878443698805217813"
},
"StudyInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.10054594808258796019777812429"
},
"CompletionFlag": "COMPLETE",
"VerificationFlag": "UNVERIFIED",
"ImagingMeasurementReport": [
  {
    "@tmr": "DCMR",
    "@tid": "1500"
  },
  {
    "LanguageOfContentItemAndDescendants": [
      "English",
      {
        "CountryOfLanguage": "UnitedStates"
      }
    ],
    "PersonObserverName": "accomplished_peafowl"
  },
  {
    "ProcedureReported": "CTAbdomen"
  },
  {
    "ImageLibrary": [
      {
        "ImageLibraryGroup": [
          {
            "": [
              "1.2.840.10008.5.1.4.1.1.2",
              "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"
            
          
        
      
    
  ]
  
}
"ImagingMeasurements": [
  {
    "MeasurementGroup": [
      {
        "TrackingIdentifier": "5b6eb4301d3175942d29985a3d0fbb00"
      },
      {
        "TrackingUniqueIdentifier": "1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1"
      },
      {
        "FindingSite": "Kidney"
      },
      {
        "Length": [
          "66.43856134",
          "mm",
          {
            "": [
              "POLYLINE",
              [
                172.83535766601562,
                270.0640869140625,
                133.79888916015625,
                343.0453186035156
              ],
              [
                "1.2.840.10008.5.1.4.1.2",
                "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"
              ]
            ]
          }
        }
      }
    ]
  },
The Patient-Study Aware System then transforms the JSON representation into the traditional binary DICOM representation and
transmits the persistent object to the PACS using either a DIMSE C-STORE Operation or a DICOMweb Store (STOW-RS) Transaction.

For clarity, the necessary Business Names Files accompanying the communication between each stage of the pipeline have been
elided from the example above. Whether particular Business Names are assumed in the hypothetical transactions between successive
steps or are explicitly communicated in Business Names Files is not defined. For the sake of argument, the simplest Business Names
File to support the initial communication between the AI Algorithm and the Lesion Manager would be as follows:

```json
[
  { "mm": {
    "@cv": "mm",
    "@csd": "UCUM",
    "@cm": "mm"
  }
  ,
  { "MeasurementGroup": {
    "@cv": "125007",
    "@csd": "DCM",
    "@cm": "Measurement Group",
    "@vt": [ "CONTAINER" ],
    "@rel": [ "CONTAINS" ]
  }
  ,
  { "Length": {
    "@cv": "410668003",
    "@csd": "SCT",
    "@cm": "Length",
    "@vt": [ "NUM" ],
    "@rel": [ "CONTAINS" ]
  }
}
]
```

For the last step in this example, conversion by a separate tool from a complete JSON representation to the binary DICOM form of
the Structured Report, the complete Business Names File (as defined for the similar example in PS3.23 Section B.3.4.2.4), as well
as a dictionary of DICOM Standard Data Element Keywords, would be required,
PS3.23

DICOM PS3.23 20xx - Alternative Representations

DICOM Standards Committee

Copyright © 2019 NEMA
Notice and Disclaimer

The information in this publication was considered technically sound by the consensus of persons engaged in the development and approval of the document at the time it was developed. Consensus does not necessarily mean that there is unanimous agreement among every person participating in the development of this document.

NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers and/or seeks out the views of persons who have an interest in the topic covered by this publication. While NEMA administers the process and establishes rules to promote fairness in the development of consensus, it does not write the document and it does not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in its standards and guideline publications.

NEMA disclaims liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. NEMA disclaims and makes no guaranty or warranty, expressed or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. NEMA does not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, NEMA is not undertaking to render professional or other services for or on behalf of any person or entity, nor is NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

NEMA has no power, nor does it undertake to police or enforce compliance with the contents of this document. NEMA does not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety-related information in this document shall not be attributable to NEMA and is solely the responsibility of the certifier or maker of the statement.
Foreword

This DICOM Standard was developed according to the procedures of the DICOM Standards Committee.

The DICOM Standard is structured as a multi-part document using the guidelines established in [ISO/IEC Directives, Part 3].
1 Scope and Field of Application

This part of the DICOM Standard specifies the alternative representations of DICOM encoded instances and messages to the traditional binary encoding defined in PS3.5.

It includes:

- Encoding of DICOM instances and messages at the Attribute level in XML
- Encoding of DICOM instances and messages at the Attribute level in JSON
- Encoding of DICOM Structured Reports at the Content Item level in JSON
2 Normative and Informative References

The following standards contain provisions that, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibilities of applying the most recent editions of the standards indicated below.

2.1 International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC)


2.2 Internet Engineering Task Force (IETF) and Internet Assigned Names Authority (IANA)


2.3 Other References

3 Definitions

For the purposes of this Standard the following definitions apply.

3.1 Codes and Controlled Terminology Definitions:

The following definitions are commonly used in this Part of the DICOM Standard:

- **Coding Schemes**: Dictionaries (lexicons) of concepts (terms) with assigned codes and well defined meanings.
- **Content Item**: A node in the Content Tree of a DICOM SR document, consisting of either a container with a coded Concept Name, or a name-value pair with a coded Concept Name and a Concept Value.
- **Content Tree**: The tree of Content Items of a DICOM SR document.
- **Context Group**: A set of coded concepts defined by a Mapping Resource forming a set appropriate to use in a particular context.
- **Context ID (CID)**: Identifier of a Context Group.
- **Template**: A pattern that describes the Content Items, Value Types, Relationship Types and Value Sets that may be used in part of a Structured Report content tree, or in other Content Item constructs, such as Acquisition Context or Protocol Context. Analogous to a Module of an Information Object Definition.
- **Template ID (TID)**: Identifier of a Template.

3.1 Representation Conversion Definitions:

The following definitions are commonly used in this Part of the DICOM Standard:

- **Business Name**: Identifier for a Concept or Attribute that corresponds to a business requirement for information exchange.

  **Note**

  See also a similar definition in PS3.20 in the context of HL7 CDA templates, and discussion in PS3.20 Section 5.2.1, which includes the statement that "the use of readable and intuitive Business Names provides a method of direct access to insert data that is specific to each clinical report instance".

- **Composite Context**: Those Attributes of higher-level Entities in the Information Model that provide the Context for newly created lower-level Entities, and which have the same values as other lower-level Entities have for the same higher-level Entities.

  **Note**

  Typically the Patient and Study Composite Context are merged with (shared by) new Series and Instance level information when a new Series of Instances is created on a different device or on a different occasion than the earlier Instances.
4 Symbols and Abbreviations

The following symbols and abbreviations are used in this Part of the Standard.

- **DICOM** Digital Imaging and Communications in Medicine
- **IOD** Information Object Definition
- **ISO** International Standards Organization
- **JSON** JavaScript Object Notation
- **NEMA** National Electrical Manufacturers Association
- **SR** Structured Reporting
- **UCUM** Unified Code for Units of Measure
- **UID** Unique Identifier
- **XML** Extensible Markup Language
- **XSLT** Extensible Stylesheet Language Transformations
5 Conventions

Terms listed in Section 3 Definitions are capitalized throughout the document.
A XML Encoding

A.1 Introduction to XML

XML ... It is .... It is described in detail by the W3C [XML].

A.2 XML Encoding of DICOM Instances

Include contents of PS3.19 Section A.1 Native DICOM Model, renaming "Native DICOM Model" to "XML Encoding of DICOM Instances in Native Format" and renumbering sections appropriately.

A.2.1 DICOM XML Encoding in Native Format

A.2.1.1 Usage

...

A.2.1.2 Identification

...

A.2.1.3 Support

...

A.2.1.4 Information Model

...

A.2.1.5 Description

...

A.2.1.6 Schema

...

A.2.1.7 Examples

...
B JSON Encoding

B.1 Introduction to JavaScript Object Notation (JSON)

JSON is a text-based open standard, derived from JavaScript, for representing data structures and associated arrays. It is language-independent, and primarily used for serializing and transmitting lightweight structured data over a network connection. It is described in detail by the Internet Engineering Task Force (IETF) in [RFC4627].

B.2 JSON Encoding of DICOM Instances and Messages

B.2.1 Introduction

The JSON Encoding of DICOM Instances and Messages defines a representation of binary-encoded DICOM SOP Instances as JSON that allows a sender or recipient of data to create or navigate through a DICOM Data Set using JSON-based tools instead of relying on tool kits that understand the binary encoding of DICOM.

B.2.2 DICOM JSON Encoding

Include contents of PS3.18 Section F.2 DICOM JSON Model, renaming "DICOM JSON Model" to "JSON Encoding of DICOM Instances and Messages" and renumbering sections appropriately:

B.2.3 Transformation to and from other DICOM Encodings

Include contents of PS3.18 Section F.3 Transformation with other DICOM Formats, renaming "DICOM JSON Model" to "JSON Encoding of DICOM Instances and Messages" and renumbering sections appropriately:

B.2.4 DICOM JSON Encoding Example

Include contents of PS3.18 Section DICOM JSON Model Example, renaming "DICOM JSON Model" to "JSON Encoding of DICOM Instances and Messages" and renumbering sections appropriately:

B.3 JSON Encoding of Structured Reports

B.3.1 Introduction

The JSON Encoding of DICOM Structured Reports defines a representation of binary-encoded DICOM Structured Report Instances as JSON that allows a sender or recipient of data to create or navigate through a DICOM Structured Report using JSON-based tools instead of relying on tool kits that understand the binary encoding of DICOM.

B.3.2 DICOM JSON Structured Report Encoding

The JSON SR encoding consists of two files:

- the JSON encoded SR Content File
- the JSON encoded Business Names File

The Content File consists of:

- a single top-level array containing a single JSON object (i.e., a single "result" in Section B.2.2 DICOM JSON Encoding terminology),
- that single JSON object containing an unordered set of subordinate JSON objects, each of which is either:
  - a JSON encoded DICOM Attribute of the top level Data Set, or
  - the root node of a JSON encoded DICOM Structured Report Content Tree
Note

In Section B.2.2 DICOM JSON Encoding, the set of subordinate objects is defined to be ordered by their property name in ascending order. No such order is required in the representation defined here, since the property names are Business Names, not hexadecimal numeric representations of a DICOM Tag. There is no need to sort the property names alphabetically, and it would be unnecessarily burdensome to the author of the JSON representation to require them to be sorted in their binary DICOM Tag order, though such sorting will be required when converting to binary DICOM encoding.

Business Names are used to identify:

- DICOM Attributes (rather than using DICOM Data Element Tags)
- Codes used in the DICOM Structured Report Content Tree

Standard DICOM Attributes used in the SR Content File are identified by the Keyword used in the PS3.6 Table 6-1 Registry of DICOM Data Elements.

Note

It is not necessary to describe Standard DICOM Attributes used in the Business Names File but it is not prohibited.

Private DICOM Attributes used in the SR Content File may be described in the Business Names File. If Private DICOM Attributes are present, corresponding Private Creator Data Elements shall also be present.

All codes used in the SR Content File shall be described in the Business Names File.

Note

Business names are not used for Code Sequence Items in Code Sequence Attributes in the top level Data Set; rather, they are encoded in the traditional manner, i.e., as individual DICOM Attributes. The reasons for this are (a) to align the DICOM Attribute header as closely with Section B.2.2 DICOM JSON Encoding as possible, and (b) very few, if any, Code Sequence Items are used in DICOM Structured Reporting SOP Classes.

The Business Names File also encodes other information related to the Content Items for which a Code is used as the Concept Name, including:

- Value Type
- Relationship

B.3.2.1 Attribute Encoding

Each DICOM Attribute in the top level Data Set, and all of the DICOM Attributes nested within Sequence Attributes in the top level Data Set, except the Attributes describing the root node of the Structured Report Content Tree, are encoded as follows:

- Each Attribute shall be encoded in the same manner as used for the JSON Encoding of DICOM Instances and Messages, as defined in Section B.2.2 DICOM JSON Encoding, except that
- In place of the eight character uppercase hexadecimal representation of a DICOM Tag used as the name of each Attribute object, one of the following shall be used:
  - a Standard DICOM Data Element Keyword from PS3.6 Table 6-1 Registry of DICOM Data Elements, or
  - a Business Name defined in the Business Names File
- The Value Representation ("vr") may be omitted for Standard Data Element Keywords (since a dictionary is expected to be available to the parser), or if it defined in the Business Names File
- A single JSON String may be used in place of the JSON Object and its enclosed "Value" Array when the value consists of a single value and the "vr" has been omitted

For example, any of the following is a valid encoding of the same Attribute that contains a single value:
The following are valid encodings of the same Attribute that contains no value (is zero length):

```
"00080020": { "vr": "DT" }

"StudyDate": { "vr": "DT" }

"StudyDate": {}
```

**Note**
For consistency with the JSON encoding described in Section B.2.2 DICOM JSON Encoding, a null value is not used except when a multi-valued attribute has one or more empty values, in which case a "Value" Array is always present.

The following would also be valid, if the Business Names "FechaDeEstudio" or "検査日" were defined in the Business Names File:

```
"FechaDeEstudio": "20130409"

"検査日": "20130409"
```

The following is an example of a Private Data Element, together with the required Private Creator, encoded using the hexadecimal tag:

```
"00190010": { "vr": "LO", "Value": [ "ACME CORP ELEMENTS" ] },
"00191001": { "vr": "US", "Value": [ "3" ] }
```

or encoded using a Business Name, if "NumberOfPhases" and "AcmeCorpCreator" were defined in the Business Names File:

```
"AcmeCorpCreator": [ "Value": [ "ACME CORP ELEMENTS" ] ],
"NumberOfPhases": [ "Value": [ "3" ] ]
```

The Attributes in the top level Data Set describing the root node of the Structured Report Content Tree that are not encoded are:

- ContentSequence
- ValueType
- ConceptNameCodeSequence
- ContinuityOfContent
- ContentTemplateSequence
- MappingResource
- TemplateIdentifier
B.3.2.2 Structured Report Content Tree Encoding

A DICOM Structured Report Content Tree consists of a nested set of Content Items of unlimited depth, beginning with a single root Content Item.

B.3.2.2.1 Content Item Encoding

Each Content Item, including the root Content Item, shall be encoded as a single JSON object, consisting of:

- a JSON object name that is a Business Name defined in the Business Names File for the DICOM Concept Name Code Sequence
- a JSON object value that is a JSON Array (or in the case of an unannotated leaf node, a JSON String), whose encoding depends on:
  - the DICOM Content Item Value Type
  - whether the Content Item is a leaf node of the tree or has children
  - whether children are by-value or by-reference

Neither the Value Type nor the Relationship Type are explicitly encoded in the JSON SR Content Tree representation, since:

- appropriate types are defined in the Business Name File
- for anonymous Content Items (those without a Concept Name), sufficient context allows the Value Type and the Relationship Type to be deduced

The following is an example of a leaf Content Item:

```
{
    "TrackingIdentifier": ["5b6eb4301d3175942d29985a3d0fbb00"]
}
```

The enclosing JSON Array may be omitted for leaf node with a single value, no children and no annotations, and a single JSON String used to represent the text value or a coded value Business Name.

The following is an example of a leaf Content Item without the enclosing JSON Array:

```
{
    "TrackingIdentifier": "5b6eb4301d3175942d29985a3d0fbb00"
}
```

B.3.2.2.2 Nested Content Encoding

For Content Items with children, the last entry of the JSON object value Array is itself an Array, which contains the ordered list of child Content Items, each of which is a JSON object.

**Note**

1. Leaf Content Items have no such final Array, rather than an empty Array.
2. Use of an Array allows for preservation of the order of child Content Items, which may be significant.
3. Use of an Array allows for multiple children with the same Concept Name, since JSON does not permit multiple JSON Objects with the same name.

The following is an example of a CODE Content Item with one child that is also a CODE Content Item:
B.3.2.2.3 Content Item Annotations

Though most of the Content Tree encoding is addressed positionally or implicitly, there are less commonly used Attributes of Content Items that need to be encoded either to preserve the full fidelity of the content or to address structural concerns, such as by-reference relationships. These concerns are addressed by the use of Standard Content Item Annotations.

The first item of the JSON object value Array may be a JSON Object, containing a set of JSON Objects each of which is a Content Item Annotation.

The names of all Standard Content Item Annotations begin with the "@" symbol.

The Standard Content Item Annotations are:

- @label  The label of a Content Item that is the target of a by-reference relationship
- @ref    The reference to a labeled Content Item that is the target of a by-reference relationship
- @tid    The TemplateIdentifier value of a CONTAINER Content Item
- @tmr    The TemplateMappingResource value of a CONTAINER Content Item
- @cont   The ContinuityOfContent value of a CONTAINER Content Item

Other annotations than these are permitted in the Content Item Annotation object, as long as their names do not begin with the "@" symbol.

Note

The intent of allowing other annotations is to allow preservation of private DICOM Attributes that may be associated with the Content Item, but no standard representation for such private information is defined. I.e., this is not a generic solution for including unrecognized private Attributes attached to Content Items.

This is an example of Content Item Annotations used to describe the template used for a root level CONTAINER (child Array illustrated but children omitted):

```json
"ImagingMeasurementReport": [
  {
    "@tmr": "DCMR",
    "@tid": "1500"
  },
  [ ... ]
]
```
B.3.2.2.4 Encoding of By-Reference Relationships

Most commonly, the Content Tree is strictly hierarchical (i.e., a tree) and many templates and some SR Storage SOP Classes constrain the encoding to that pattern. However, by-reference relationships (which allow for a directed acyclic graph) are permitted by the underlying mechanism and hence are supported in the JSON encoding by use of Content Item Annotations.

Both the referenced Content Item and the referencing Content Item need to be decorated with Content Item Annotations as follows:

- The referenced Content Item must have an @label annotation, whose value shall be a string unique amongst such labels within the instance.
- The referencing Content Item must have an @ref annotation, whose value shall correspond to the @label annotation of a Content Item within the instance.

Note

1. The @label and @ref annotations are not required to be the numeric hierarchical position in the Content Tree, in order to simplify the creator's task.

2. In the traditional binary DICOM SR encoding, the references are made using ReferencedContentItemIdentifier (see PS3.3 C.17.3.2.5), which is the numeric hierarchical position (the set of ordinal positions along the by-value relationship path from the root Content Item) in the Content Tree, so the parser of the JSON representation is required to map the @ref values to the numeric hierarchical position by tracking the position of Content Items with @label annotations. Since forward references are permitted, this may require two passes of the JSON file.

The following is a simplified example of a reference from a CAD finding SCOORD to an IMAGE Content Item in an Image Library:

```json
[  
  {"ImageLibrary": [['': [  
    {'@label': 'label1'},  
    "1.2.840.10008.5.1.4.1.1.1.2",  
    "1.3.6.1.4.1.5962.99.1.993064428.2122236180.1358202762732.2.0"
  ]],},  
  {"CADProcessingAndFindingsSummary": [  
    "AllAlgorithmsSucceededWithFindings",  
    [{"IndividualImpressionRecommendation": [{  
      "Center": [  
        "POINT",
        [165,2433],
        [{"": [{"@ref": "label1"}]}
      ],
      ...  
    }]}]
  ]}
]
```

B.3.2.2.5 Encoding of Content Items of Specific Value Type

The encoding of Content Items depends on their Value Type. There is a specific JSON representation for each of the Value Types defined in PS3.3 Table C.17-5 Document Content Macro Attributes.

Note

The pattern of encoding for each Value Type not only allows each Content Item to be encoded with full fidelity, but also allows for recognition of the Value Type by the parser when it is not explicitly defined for the Concept Name in the Business Names File, or is potentially ambiguous, such as for anonymous Content Items that do not have a Concept Name.
B.3.2.2.5.1 Encoding of Content Items Without a Value

The following Value Type never has a value, though may have children, and is encoded in the JSON value Array without a value:

- CONTAINER

The following is an example of a CONTAINER Content Item, where the code and Value Type for "ImageLibrary" are defined in the Business Names File:

```json
{
    "ImageLibrary": []
}
```

The following is an example of a CONTAINER Content Item, with one child that is a CONTAINER with no children of its own:

```json
{
    "ImageLibrary": [
        {
            "ImageLibraryGroup": []
        }
    ]
}
```

B.3.2.2.5.2 Encoding of Content Items with a Single Value

The following Value Types consist of a single value that is either textual or a code, and are all encoded in the JSON value Array using a single JSON String:

- CODE
- DATE
- DATETIME
- PNAME (TBD. Open Issue: should be decomposed as for DICOM Attribute PN VR ?)
- TEXT
- TIME
- UIDREF

The JSON String value may represent a text value or a coded value Business Name, depending on the Value Type deduced from the Concept Name.

The following are examples of a TEXT Content Item with no children, where the code and Value Type for "Comment" is defined in the Business Names File, encoded with and without the enclosing JSON Array:

```json
{
    "Comment": [
        "Kidney"
    ]
}
```

```json
{
    "Comment": "Kidney"
}
```
The following is an example of a CODE Content Item with no children, where the code and Value Type for "FindingSite", as well as the code for the "Kidney", are defined in the Business Names File, encoded with and without the enclosing JSON Array:

```
{
  "FindingSite": [ 
    "Kidney"
  ]
}
{
  "FindingSite": "Kidney"
}
```

The following is an example of a CODE Content Item with one child that is also a CODE Content Item, with no children of its own:

```
{
  "LanguageOfContentItemAndDescendants": [ 
    "English",
    [ 
      { 
        "CountryOfLanguage": "UnitedStates"
      }
    ]
  ]
}
```

### B.3.2.2.5.3 Encoding of Numeric Content Items

The following Value Type consists of a single numeric value and its measurement units encoded in the JSON value Array as a pair of JSON Strings:

- NUM

The first JSON String is the value of the DS VR NumericValue in MeasuredValueSequence. [TBD. Should this be a Number per discussions re. PS3.18 JSON ????]

The second JSON String is the Business Name of the code in the MeasurementUnitsCodeSequence in MeasuredValueSequence.

The following is an example of a NUM Content Item with no children, where the code and Value Type for "Length", as well as the code for the "mm", are defined in the Business Names File:

```
{
  "Length": [ 
    "66.43856134",
    "mm"
  ]
}
```

The encoding of the other less frequently used Attributes within MeasuredValueSequence defined in PS3.3 Table C.18.1-1 Numeric Measurement Macro Attributes is addressed by the use of Content Item Annotations as follows:

- TBD.

### B.3.2.2.5.4 Encoding of Content Items That Reference Storage SOP Instances

The following Value Types reference Storage SOP Instances and encode in the JSON value Array the SOPClassUID and SOPInstanceUID of the referenced instance as a pair of JSON Strings:
The first JSON String is the value of the ReferencedSOPClassUID in ReferencedSOPSequence.

The second JSON String is the value of the ReferencedSOPInstanceUID in MeasuredValueSequence.

The following is an example of an IMAGE Content Item with no children, for which there is no Concept Name encoded (i.e., is an anonymous Content Item):

```
{
  "": ["1.2.840.10008.5.1.4.1.1.2",
       "1.3.6.1.4.1.14519.5.2.1.9203.4004.2680184228881857326516023762"
  ]
}
```

**Note**

A parser can detect that this is an IMAGE Content Item even in the absence of a Business Name for the Concept Name by recognizing that the JSON value Array contains two JSON Strings, the first of which is a recognized Storage SOP Class UID. Were the reference to be to an unrecognized SOP Class, such as a Private SOP Class, the risk of ambiguity can be avoided by providing a Concept Name with the appropriate Value Type even if the Template does not require it. TBD. Open Issue: allow annotation with explicit Value Type for such cases?

If an IMAGE Content Item has other Attributes than ReferencedSOPClassUID and ReferencedSOPInstanceUID, they are encoded as additional values in the following order:

1. Array of ReferencedFrameNumber
2. ReferencedSegmentNumber
3. ReferencedSOPClassUID in ReferencedSOPSequence (to a Presentation State Instance)
4. ReferencedSOPInstanceUID in ReferencedSOPSequence (to a Presentation State Instance)
5. ReferencedSOPClassUID in ReferencedRealWorldValueMappingInstanceSequence
6. ReferencedSOPInstanceUID in ReferencedRealWorldValueMappingInstanceSequence

Trailing unused values may be elided but intervening values are required to be null if there is no value, in order to preserve the positional order.

**Note**

These additional Attributes are encoded positionally rather than as named Content Item Annotations for simplicity and compactness.

The following is an example of an IMAGE Content Item with no children, for which there is no Concept Name encoded (i.e., is an anonymous Content Item) and with a ReferencedSegmentNumber:

```
{
  "": ["1.2.840.10008.5.1.4.1.1.66.4",
       "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288813963226516023762",
       null,
       3
  ]
}
```
[TBD. If a WAVEFORM Content Item has other Attributes than ReferencedSOPClassUID and ReferencedSOPInstanceUID ....]

B.3.2.2.5.5 Encoding of Coordinate Content Items

The following Value Types encode in the JSON value Array the type of coordinates as a JSON String followed by a JSON Array of numeric coordinates:

• SCOORD
• SCOORD3D
• TCOORD

For SCOORD Content Items:
1. The JSON String is the value of the GraphicType.
2. The JSON Array contains the values of GraphicData

The following is an example of an SCOORD Content Item SELECTED FROM an IMAGE, for which there is no Concept Name encoded for either (i.e., they are anonymous Content Items):

```json
{
    "": [
        "POLYLINE",
        [
            172.83535766601562,
            270.0640869140625,
            133.79888916015625,
            343.0453186035156
        ],
        {
            "": [
                "1.2.840.10008.5.1.4.1.1.2",
                "1.3.6.1.4.1.14519.5.2.1.9203.4004.268018422288818573226516023762"
            ]
        }
    ]
}
```

Note
1. A parser can detect that this is an SCOORD Content Item even in the absence of a Business Name for the Concept Name by recognizing that the JSON value Array contains a JSON String followed by a JSON Array of Numbers; further the JSON String is a recognized GraphicType. The risk of ambiguity can be avoided by providing a Concept Name with the appropriate Value Type even if the Template does not require it. TBD. allow annotation with explicit Value Type for such cases?
2. A parser can assume that a SELECTED FROM Relationship Type is needed between the parent SCOORD Content Item and the child IMAGE Content Item, since that is the only relationship permitted between these two Value Types.
3. A parser can assume that an INFERRED FROM Relationship Type is needed between a parent TEXT, CODE or NUM Content Item and a child SCOORD Content Item, since that is the only relationship permitted between these Value Types.

[TBD. Add other Attributes as Annotations or positionally, specifically PixelOriginInterpretation (for WSI) and FiducialUID.]
[TBD. For SCOORD3D Content Items...]

For TCOORD Content Items:

1. The JSON String is the value of TemporalRangeType.

2. [TBD. The JSON Array contains the values of ReferencedSamplePositions, ReferencedTimeOffsets or Referenced DateTime. How to distinguish these? Need another type String?]

B.3.2.2.6 Encoding of Business Names File

The Business Names File consists of:

• a single top-level array containing zero or more JSON objects,

• each of which is a JSON object that describes either:

  • a coded concept used as the value of a name-value pair,

  • a coded concept used as the concept name of a name-value pair, or

  • a DICOM data element

Each such JSON object in the top level array contains a single subordinate JSON object that has a property name that is the Business Name

Note

The JSON objects are nested since the same property name may be used to describe both a coded concept and a DICOM data element. E.g., such concepts as "StudyDate" or "Modality" may be encoded as a DICOM Attribute or Content Item or both.

A coded concept, whether used as a value or concept name, is defined with the following basic properties:

@cv The code value of the coded concept

@csp The coding scheme designator of the coded concept

@cm The code meaning of the coded concept

This is an example of the definition of a coded concept for (41806-1, LN, "CT Abdomen"):

```
{
  "CTAbdomen": {
    "@cv": "41806-1",
    "@csd": "LN",
    "@cm": "CT Abdomen"
  }
}
```

[TBD. Add annotations for extended attributes and long or URL codes and alternative codes.]

A coded concept that is used as the Concept Name of a Content Item is defined with the following additional properties:

@vt The Value Type of the Content Item, encoded as a JSON Array of one or more JSON Strings corresponding to the Value Types defined in PS3.3 Table C.17.3-7.

@rel The Relationship Type of the Content Item with its parent Content Item, encoded as a JSON Array of one or more JSON Strings corresponding to the Relationship Types defined in PS3.3 Table C.17.3-8.
Note

A JSON Array rather than a single JSON String value is used, since some Concept Names may be used with different types in different contexts. E.g., (121050, DCM, "Equivalent Meaning of Concept Name") may have a CODE or a TEXT Value Type. (111010, DCM, "Center") may have a HAS PROPERTIES or INFERRED FROM Relationship Type.

[TBD. Do we want to allow the common degenerate case when only a single String is needed to be encoded without the surrounding Array?]

[TBD. Need to make sure ambiguity resolution rules are defined and robust for such use cases, else require annotation of @vt or @rel inline in JSON file.]

This is an example of the definition of a coded concept for (363698007, SCT, "Finding Site") that may be used as a Concept Name of a Content Item, with the Value Type of CODE and a Relationship Type of HAS CONCEPT MOD:

```
{
    "FindingSite": {
        "@cv": "363698007",
        "@csd": "SCT",
        "@cm": "Finding Site",
        "@vt": ["CODE"],
        "@rel": ["HAS CONCEPT MOD"]
    }
}
```

A DICOM data element, whether Standard or Private, is defined with the following basic properties:

@tag The eight character uppercase hexadecimal representation of a DICOM Tag, as defined in Section B.2.2 DICOM JSON Encoding.

@vr The Value Representation encoded as JSON String corresponding to the Value Representations defined in PS3.5 Table 6.2-1

In the case of Private Data Elements, the tag shall be exactly as encoded in the Data Set, i.e., with the block number included. The corresponding Private Creator Data Element is included in the JSON encoded SR Content File.

Note

This is to be consistent with the encoding defined in Section B.2.2 DICOM JSON Encoding. It also means that neither the creator nor the parser needs to be aware of the distinction between Private and Standard Data Elements when transforming between the binary and JSON representations.

This is an example of the definition of a Standard Data Element, Series Instance UID (0020,000E):

```
{
    "SeriesInstanceUID": {
        "@tag": "0020000E",
        "@vr": "UI"
    }
}
```

This is an example of the definition of a Private Data Element and its corresponding Private Creator:

```
{
    "AcmeCorpCreator": {
        "@tag": "00190010",
        "@vr": "LO"
    }
}
```
1. The value of the Private Creator Data Element is not specified in the Business Names File, and needs to be encoded in the JSON Content File, e.g., "ACME CORP ELEMENTS" or similar.

   TBD. Do we want to optimize this and add another property like @creator="ACME CORP ELEMENTS" in the business names file? Probably not, since the element still needs to be included in the JSON Content File, and it should have a value there too, so it would be redundant.

2. There is no relationship defined in the Business Names File between the Private Data Element and the Private Creator. It is only in the JSON Content File, which makes use of these Business Names, that the correspondence between the definition of a block of Private Data Elements and the use of that block is established through the Data Element Tag numerical values as defined in PS3.5 Section 7.8.1.

B.3.3 DICOM JSON Structured Report Encoding Examples

B.3.3.1 DICOM JSON Simple Single Linear Measurement Encoding Example

B.3.3.1.1 Simplest Example

The following is the simplest example of the content that TID 1500 allows for a single linear distance measurement.

Note that TID 1500 requires both a Tracking Identifier and Tracking Unique Identifier. These are minimal requirements of the Template, not of the JSON transformation per se, and might not be needed by other Templates.

B.3.3.1.1.1 Semantic Content

A compact representation of the semantic content of the transformed DICOM SR tree is shown here:

```
: CONTAINER: (126000,DCM,"Imaging Measurement Report") [SEPARATE] (DCMR,1500)
>CONTAINS: CONTAINER: (126010,DCM,"Imaging Measurements") [SEPARATE]
>>CONTAINS: CONTAINER: (125007,DCM,"Measurement Group") [SEPARATE]
>>>HAS OBS CONTEXT: TEXT: (112039,DCM,"Tracking Identifier") = "5b6eb4301d3175942d29985a3d0fbb00"
>>>HAS OBS CONTEXT: UIDREF: (112040,DCM,"Tracking Unique Identifier") = "1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1"
>>>CONTAINS: NUM: (410668003,SCT,"Length") = 66.43856134 (mm,UCUM,"mm")
>>>INFERRED FROM: SCOORD: = POLYLINE {172.835357666016,270.064086914062,133.798889160156,343.045318603516}
>>>SELECTED FROM: IMAGE: = (1.2.840.10008.5.1.4.1.1.2,1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1)
```

B.3.3.1.2 JSON Content Item Tree Only

This is the JSON File consisting of just the Content Item Tree, with the DICOM top level Data Set omitted for clarity, such as might be produced by an AI Algorithm and Lesion Manager before merging with the Composite Context:

```
[ "ImagingMeasurementReport": [
   { "@tmr": "DCMR",
     "@tid": "1500"
   },
   "ImagingMeasurements": [
```


B.3.3.1.2 More Realistic Example

The following is a more realistic example of a TID 1500 encoding of a single linear distance measurement, which adds Language and Country, the Person Observer, the Procedure Reported, an Image Library entry, and a Finding Site.

B.3.3.1.2.1 Semantic Content

A compact representation of the semantic content of the transformed DICOM SR tree is shown here:

```json
[  
  {"MeasurementGroup": [  
    [  
      "TrackingIdentifier": "5b6eb4301d3175942d29985a3d0fbb00"  
    ],  
    "TrackingUniqueIdentifier": "1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1"  
  ],  
  "Length": [  
    "66.43856134",  
    "mm",  
    {  
      "": [  
        "POLYLINE",  
        [  
          172.83535766601562,  
          270.0640869140625,  
          133.79888916015625,  
          343.0453186035156  
        ],  
        [  
          "": [  
            "1.2.840.10008.5.1.4.1.1.2",  
            "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"  
          ]  
        ]  
      ]  
    ]  
  ]},  
]  
]
```

: CONTAINER: (126000, DCM,"Imaging Measurement Report") [SEPARATE] (DCMR,1500)
> HAS CONCEPT MOD: CODE: (121049, DCM,"Language of Content Item and Descendants") = (eng,RFC5646,"English")
B.3.3.1.2.2 JSON Content Item Tree Only

This is the JSON File consisting of just the Content Item Tree, with the DICOM top level Data Set omitted for clarity, such as might be produced by an AI tool before merging with the composite context:

```json
[
  "ImagingMeasurementReport": [
    {
      "@tmr": "DCMR",
      "@tid": "1500"
    },
    [
      {
        "LanguageOfContentItemAndDescendants": [
          "English",
          [
            {
              "CountryOfLanguage": "UnitedStates"
            }
          ]
        },
        {
          "PersonObserverName": "accomplished_peafowl"
        },
        {
          "ProcedureReported": "CTAbdomen"
        },
        {
          "ImageLibrary": [
            {
              "ImageLibraryGroup": [
                [
                  {
                    "": [
                      "1.2.840.10008.5.1.4.1.1.2",
                      "1.3.6.1.4.1.14519.5.2.1.9203.4004.268018422288818573226516023762"
                    ]
                  }
                ]
              ]
            }
          ]
        }
      ]
    }
  ]
]
```
```json
{
    "StudyDate": "19870620",
    "StudyTime": "135823"
}

"ImagingMeasurements": [
    {
        "MeasurementGroup": [
            {
                "TrackingIdentifier": "5b6eb4301d3175942d29985a3d0fbb00"
            },
            {
                "TrackingUniqueIdentifier": "1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1"
            },
            {
                "FindingSite": "Kidney"
            },
            {
                "Length": [
                    "66.43856134",
                    "mm",
                    [
                        ": ["P"OLYLINE",
                        [172.83535766601562,
                        270.0640869140625,
                        133.7988916015625,
                        343.0453186035156
                        ],
                        ["1.2.840.10008.5.1.4.1.1.2",
                        "1.3.6.1.4.1.14519.5.2.1.9203.4004.268018422288818573226516023762"
                        ]
                    ]
                }
            }
        ]
    }
}
```
B.3.3.1.2.3 Entire JSON File

This is the entire JSON File consisting of the DICOM top level Data Set and the Content Item Tree, such as might be produced after merging the AI tool output with the DICOM composite context required to encode a valid SOP Instance:

```json
{
"SOPClassUID": "1.2.840.10008.5.1.4.1.1.88.22",
"SOPInstanceUID": "1.3.6.1.4.1.5962.1.1.0.0.0.1572701186.46156.1",
"StudyDate": "19870620",
"SeriesDate": {},
"ContentDate": "20171126",
"StudyTime": "135823",
"ContentTime": "224217",
"AccessionNumber": {},
"Modality": "SR",
"Manufacturer": "PixelMed",
"InstitutionName": {},
"ReferringPhysicianName": {},
"StationName": "NONE",
"StudyDescription": "Renal",
"SeriesDescription": "Crowds Cure Cancer Annotation as Measurement Report",
"ManufacturerModelName": "XSLT from annotations_expanded.csv",
"ReferencedPerformedProcedureStepSequence": {},
"PatientName": {
  "Value": [
    {
      "Alphabetic": "TCGA-BP-4343"
    }
  
  }
},
"PatientID": "TCGA-BP-4343",
"PatientBirthDate": {},
"PatientSex": {},
"DeviceSerialNumber": "9723613413261",
"SoftwareVersions": "0.1",
"StudyInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.1005459480825879601977812429",
"SeriesInstanceUID": "1.3.6.1.4.1.5962.1.3.0.0.1572701186.46156.1",
"StudyID": {},
"SeriesNumber": "4578",
"InstanceNumber": "1",
"AuthorObserverSequence": {
  "Value": [
    {
      "InstitutionName": {},
      "InstitutionCodeSequence": {},
      "PersonIdentificationCodeSequence": {},
      "ObserverType": "PSN",
      "PersonName": {
        "Value": [
      
      ]
    }
  ]
}
```

[
  
}
]
"Alphabetic": "accomplished_peafowl"

"PerformedProcedureCodeSequence": {},
"CurrentRequestedProcedureEvidenceSequence": {
  "Value": [
    {"ReferencedSeriesSequence": {
      "Value": [
        {"ReferencedSOPSequence": {
          "Value": [
            {"ReferencedSOPClassUID": "1.2.840.10008.5.1.4.1.1.2",
             "ReferencedSOPInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"
          ]
        },
        {"SeriesInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.178060436087844836988805217813"
        },
        {"StudyInstanceUID": "1.3.6.1.4.1.14519.5.2.1.9203.4004.10054948082587796019777712429"
        }
      ]
    },
    "CompletionFlag": "COMPLETE",
    "VerificationFlag": "UNVERIFIED",
    "ImagingMeasurementReport": [
      {
        "@tmr": "DCMR",
        "@tid": "1500"
      },
      {
        "LanguageOfContentItemAndDescendants": [
          "English",
          {"CountryOfLanguage": "UnitedStates"
          }
        ],
        "PersonObserverName": "accomplished_peafowl"
      },
      {"ProcedureReported": "CTAbdomen"
      },
      {"ImageLibrary": [
        "ImageLibraryGroup": [
          "ImageLibrary": [
          ]
        ]
      ]
    }
  ]
}
["1.2.840.10008.5.1.4.1.1.2",
"1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762",
{
  "Modality": "ComputedTomography"
},
{
  "StudyDate": "19870620"
},
{
  "StudyTime": "135823"
}
],
{
  "ImagingMeasurements": [
  {
    "MeasurementGroup": [
      {
        "TrackingIdentifier": "5b6eb4301d3175942d29985a3d0fbb00"
      },
      {"TrackingUniqueIdentifier": "1.3.6.1.4.1.5962.1.1.0.0.0.1535644357.22655.1"}
    ],
    "FindingSite": "Kidney"
  },
  {
    "Length": [
      "66.43856134",
      "mm"
    ],
    "": [
      "POLYLINE",
      [172.83535766601562, 270.0640869140625, 133.7988916015625, 343.0453186035156
    ],
    ["": [
      "1.2.840.10008.5.1.4.1.1.2",
      "1.3.6.1.4.1.14519.5.2.1.9203.4004.26801842288818573226516023762"
    ]
  }
}
B.3.3.1.2.4 JSON Business Names File

This is the JSON Business Names File for this example, which defines the coded concepts used, as well as the Value Type and Relationship Type for those coded concepts used as Concept Names for Content Items:

```json
[
    {
        "mm": {
            "@cv": "mm",
            "@csd": "UCUM",
            "@cm": "mm"
        }
    },
    {
        "ImagingMeasurementReport": {
            "@cv": "126000",
            "@csd": "DCM",
            "@cm": "Imaging Measurement Report",
            "@vt": [
                "CONTAINER"
            ],
            "@rel": [
                "HAS OBS CONTEXT"
            ]
        }
    },
    {
        "TrackingIdentifier": {
            "@cv": "112039",
            "@csd": "DCM",
            "@cm": "Tracking Identifier",
            "@vt": [
                "TEXT"
            ],
            "@rel": [
                "HAS OBS CONTEXT"
            ]
        }
    },
    {
        "PersonObserverName": {
            "@cv": "121008",
            "@csd": "DCM",
            "@cm": "Person Observer Name",
            "@vt": [
                "TEXT"
            ],
            "@rel": [
                "HAS OBS CONTEXT"
            ]
        }
    }
]
```
"@vt": [
  "PNAME"
],
"@rel": [
  "HAS OBS CONTEXT"
]
}
",
{
"CTAbdomen": {
  "@cv": "41806-1",
  "@csd": "LN",
  "@cm": "CT Abdomen"
}
},
{
"StudyDate": {
  "@cv": "111060",
  "@csd": "DCM",
  "@cm": "Study Date",
  "@vt": [
    "DATE"
  ],
  "@rel": [
    "HAS ACQ CONTEXT"
  ]
}
},
{
"FindingSite": {
  "@cv": "363698007",
  "@csd": "SCT",
  "@cm": "Finding Site",
  "@vt": [
    "CODE"
  ],
  "@rel": [
    "HAS CONCEPT MOD"
  ]
}
},
{
"MeasurementGroup": {
  "@cv": "125007",
  "@csd": "DCM",
  "@cm": "Measurement Group",
  "@vt": [
    "CONTAINER"
  ],
  "@rel": [
    "CONTAINS"
  ]
}
},
{
"ProcedureReported": {
  "@cv": "121058",
  "@csd": "DCM",
  "@cm": "Procedure reported",
  "@vt": [
"CODE",
"@rel": [
  "HAS CONCEPT MOD"
]
},

"Kidney": {
  "@cv": "64033007",
  "@csd": "SCT",
  "@cm": "Kidney"
}
},

"StudyTime": {
  "@cv": "111061",
  "@csd": "DCM",
  "@cm": "Study Time",
  "@vt": [
    "TIME"
  ],
  "@rel": [
    "HAS ACQ CONTEXT"
  ]
}
},

"English": {
  "@cv": "eng",
  "@csd": "RFC5646",
  "@cm": "English"
}
},

"ImageLibraryGroup": {
  "@cv": "126200",
  "@csd": "DCM",
  "@cm": "Image Library Group",
  "@vt": [
    "CONTAINER"
  ],
  "@rel": [
    "CONTAINS"
  ]
}
},

"TrackingUniqueIdentifier": {
  "@cv": "112040",
  "@csd": "DCM",
  "@cm": "Tracking Unique Identifier",
  "@vt": [
    "UIDREF"
  ],
  "@rel": [
    "HAS OBS CONTEXT"
  ]
}
},

"@rel": [ 
  "HAS CONCEPT MOD"
]


```json
{
    "CountryOfLanguage": {
        "@cv": "121046",
        "@csd": "DCM",
        "@cm": "Country of Language",
        "@vt": [
            "CODE"
        ],
        "@rel": [
            "HAS CONCEPT MOD"
        ]
    }
},
{
    "LanguageOfContentItemAndDescendants": {
        "@cv": "121049",
        "@csd": "DCM",
        "@cm": "Language of Content Item and Descendants",
        "@vt": [
            "CODE"
        ],
        "@rel": [
            "HAS CONCEPT MOD"
        ]
    }
},
{
    "Length": {
        "@cv": "410668003",
        "@csd": "SCT",
        "@cm": "Length",
        "@vt": [
            "NUM"
        ],
        "@rel": [
            "CONTAINS"
        ]
    }
},
{
    "ImagingMeasurements": {
        "@cv": "126010",
        "@csd": "DCM",
        "@cm": "Imaging Measurements",
        "@vt": [
            "CONTAINER"
        ],
        "@rel": [
            "CONTAINS"
        ]
    }
},
{
    "ImageLibrary": {
        "@cv": "111028",
        "@csd": "DCM",
        "@cm": "Image Library",
        "@vt": [
            "CONTAINER"
        ],
        "@rel": [
            "CONTAINS"
        ]
    }
}
```
"@rel": [ 
  "CONTAINS"
],

"Modality": {
  "@cv": "121139",
  "@csd": "DCM",
  "@cm": "Modality",
  "@vt": [
    "CODE"
  ],
  "@rel": [
    "HAS ACQ CONTEXT"
  ]
},

"ComputedTomography": {
  "@cv": "CT",
  "@csd": "DCM",
  "@cm": "Computed Tomography"
},

"UnitedStates": {
  "@cv": "US",
  "@csd": "ISO3166_1",
  "@cm": "United States"
}
F DICOM JSON Model

<table>
<thead>
<tr>
<th>Amend PS3.18 as follows (changes to existing text are bold and underlined for additions and struckthrough for removals):</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.1 Introduction to JavaScript Object Notation (JSON)</td>
</tr>
</tbody>
</table>

JSON is a text-based open standard, derived from JavaScript, for representing data structures and associated arrays. It is language-independent, and primarily used for serializing and transmitting lightweight structured data over a network connection. It is described in detail by the Internet Engineering Task Force (IETF) in [RFC4627], available at http://www.ietf.org/rfc/rfc4627.txt.

See Section B.2.1 “Introduction”.

The DICOM JSON Model complements the XML-based Native DICOM Model, by providing a lightweight representation of data returned by DICOM web services. While this representation can be used to encode any type of DICOM Data Set it is expected to be used by client applications, especially mobile clients, such as described in the QIDO-RS use cases (see Annex HHH “Transition from WADO to RESTful Services (Informative)” in PS3.17).

F.2 DICOM JSON Model

Retired. See Section B.2.2 “DICOM JSON Encoding”.

F.3 Transformation with other DICOM Formats

Retired. See Section B.2.3 “Transformation to and from other DICOM Encodings”.

F.4 DICOM JSON Model Example

Retired. See Section B.2.4 “DICOM JSON Encoding Example”.

- Draft -
A Data Exchange Models

Amend PS3.19 as follows (changes to existing text are bold and *underlined* for additions and *struckthrough* for removals):

A.1 Native DICOM Model

A.1.1 Usage

*Retired. See Section A.2.1.1 “Usage”.*

A.1.2 Identification

*Retired. See Section A.2.1.2 “Identification”.*

A.1.3 Support

*Retired. See Section A.2.1.3 “Support”.*

A.1.4 Information Model

*Retired. See Section A.2.1.4 “Information Model”.*

A.1.5 Description

*Retired. See Section A.2.1.5 “Description”.*

A.1.6 Schema

*Retired. See Section A.2.1.6 “Schema”.*

A.1.7 Examples

*Retired. See Section A.2.1.7 “Examples”.*
A Structured Reporting Templates (Normative)

Amend PS3.16 as follows (changes to existing text are bold and underlined for additions and struckthrough for removals):

TID 1500 Measurement Report

<table>
<thead>
<tr>
<th>NL</th>
<th>Rel with Parent</th>
<th>VT</th>
<th>Concept Name</th>
<th>VM</th>
<th>Req Type</th>
<th>Condition</th>
<th>Value Set Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>CONTAINER</td>
<td>DCID 7021 &quot;Measurement Report Document Titles&quot;</td>
<td>1</td>
<td>M</td>
<td>Root node</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>&gt; HAS CONCEPT MOD</td>
<td>INCLUDE</td>
<td>DTID 1204 &quot;Language of Content Item and Descendants&quot;</td>
<td>1</td>
<td>MU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&gt; HAS OBS CONTEXT</td>
<td>INCLUDE</td>
<td>DTID 1001 &quot;Observation Context&quot;</td>
<td>1</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&gt; HAS CONCEPT MOD</td>
<td>CODE</td>
<td>EV (121058, DCM, &quot;Procedure reported&quot;)</td>
<td>1-n</td>
<td>MU</td>
<td>BCID 100 &quot;Quantitative Diagnostic Imaging Procedures&quot;</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>&gt; CONTAINS</td>
<td>INCLUDE</td>
<td>DTID 1600 &quot;Image Library&quot;</td>
<td>1</td>
<td>MU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>&gt; CONTAINS</td>
<td>CONTAINER</td>
<td>EV (126010, DCM, &quot;Imaging Measurements&quot;)</td>
<td>1</td>
<td>C</td>
<td>IF row 10 and 12 are absent</td>
<td></td>
</tr>
</tbody>
</table>