

# ***DICOM Structured Reporting***



# ***DICOM Structured Reporting***

**David A. Clunie**

**PixelMed Publishing**

**Bangor, Pennsylvania**

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind, and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

Copyright © 2000 by PixelMed Publishing.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

Library of Congress Card Number: 00-191700

ISBN 0-9701369-0-0

*For Eleanor ...*



---

---

<i>Who This Book is For</i>	21
<i>Outline of This Book</i>	22
<i>Versions of the Standard</i>	24
<i>Conventions Used in this Book</i>	24
<i>Acknowledgements</i>	26
<i>Feedback</i>	27

## **CHAPTER 1**

### **Structured Reporting Concepts 29**

<i>Not Just Reports ...</i>	30
<i>Semantics not presentation</i>	30
<i>Constraints, IODs and Templates</i>	31
<i>Content</i>	32
<i>Trees and Relationships</i>	33
<i>Coded Concepts and Values</i>	34
<i>Living Without Codes</i>	35
<i>Report Management</i>	36
<i>Summary</i>	38

## **CHAPTER 2**

### **Codes and Controlled Terminology 39**

<i>Basic Concepts of Coded Entries</i>	39
<i>DICOM Encoding</i>	40
<i>Coding Scheme Designator and Version</i>	42
<i>Pros and Cons of Codes</i>	43
<i>Coding Schemes to Choose From</i>	44
<i>Codes for Structured Reporting</i>	49
<i>What does a Code Represent?</i>	51
<i>Code Meaning Revisited</i>	53
<i>Not Otherwise Specified</i>	54
<i>Context Groups and Mapping Resources</i>	55
<i>Context Groups and Synonyms</i>	57
<i>Beyond Triplets... Enhanced Encoding Mode</i>	60
<i>Private and Extended Coding Schemes and Mapping Resources</i>	61
<i>Summary</i>	63

## **CHAPTER 3**

### **Content Items: Concept Names and Values 65**

<i>Document Title</i>	65
<i>Headings</i>	67
<i>Encoding Headings</i>	69

---



---

<i>Value Types</i>	70
<i>Concept Names</i>	71
<i>Value Type CONTAINER</i>	72
<i>Continuity of Content</i>	72
<i>Value Type TEXT</i>	73
<i>Value Type CODE</i>	75
<i>Value Type NUM</i>	76
<i>Value Type PNAME</i>	77
<i>Value Types DATE, TIME and DATETIME</i>	80
<i>Reference Value Types</i>	83
<i>Value Type UIDREF</i>	85
<i>Value Type COMPOSITE</i>	86
<i>Value Type IMAGE</i>	88
<i>Value Type WAVEFORM</i>	90
<i>Coordinate Value Types</i>	91
<i>Value Type SCOORD</i>	92
<i>Value Type TCOORD</i>	97
<i>Purpose of Reference</i>	100
<i>Only Simple Types</i>	102
<i>Summary</i>	103

## **CHAPTER 4**

### **Encoding & Relationships: Building the Tree** 105

<i>Encoding a Tree</i>	105
<i>Relationships</i>	110
<i>Relationship Types</i>	112
<i>Contains Relationship Type</i>	115
<i>Has Properties Relationship Type</i>	121
<i>Inferred From Relationship Type</i>	123
<i>By-Reference Relationships</i>	126
<i>Referenced Content Item Identifiers</i>	129
<i>Selected From Relationship Type</i>	130
<i>Has Observation Context Relationship Type</i>	130
<i>Has Acquisition Context Relationship Type</i>	131
<i>Has Concept Modifier Relationship Type</i>	131
<i>Summary</i>	136

## **CHAPTER 5**

### **Images, Waveforms, and Voice Audio** 137

<i>Report+Images</i>	137
<i>The Limits of the Web Paradigm</i>	138
<i>Image References</i>	139
<i>Presentation States</i>	141

---

---

<i>Spatial Coordinates</i>	144
<i>Temporal Coordinates</i>	145
<i>Waveforms</i>	148
<i>Living with Voice</i>	149
<i>Living With Paper</i>	151
<i>Scanned Documents</i>	152
<i>Scanned Images</i>	152
<i>Drawings</i>	153
<i>Forms</i>	154
<i>Acquisition Context</i>	155
<i>Acquisition Context in SR</i>	158
<i>Summary</i>	160

## **CHAPTER 6**

### ***SOP Classes and Conformance*** 161

<i>IODs, SOP Classes and Conformance</i>	161
<i>SR IODs and SOP Classes</i>	162
<i>What does conformance actually mean?</i>	164
<i>Conformance and Negation Related Issues</i>	166
<i>Negation using Pre-coordinated Concept Names</i>	166
<i>Negation using Concept Modifiers</i>	167
<i>Observation Context</i>	168
<i>Conformance and References to Other SOP Instances</i>	168
<i>Other Conformance Issues</i>	169
<i>Summary</i>	169

## **CHAPTER 7**

### ***Context: Who did what to whom, when...*** 171

<i>Observer Context</i>	172
<i>Subject Context</i>	173
<i>Procedure Context</i>	174
<i>Identification vs. Description</i>	174
<i>Encoding Observation Context</i>	175
<i>Inheritance of Observation Context</i>	177
<i>Initial Context</i>	181
<i>Quoted Content</i>	182
<i>Not Observation Context</i>	185
<i>Summary</i>	185

---

---

**CHAPTER 8**

**Document Management: SR in the  
Real World 187**

*Instances 187*  
*Unique Identifiers 187*  
*"Document" Paradigm: Persistence vs.  
Transience 189*  
*Copies and Versions 191*  
*States: Completion and Verification 195*  
*Current and Other Evidence: References  
Revisited 198*  
*Request and Procedure Information 202*  
*SR Document General Module 203*  
*Example of Management Information  
Encoding 203*  
*SR Document Series Module 207*  
*Query and Retrieval 208*  
*Query Information Models 210*  
*Query Keys: Matching and Return 212*  
*Query Keys for SR 214*  
*Workflow Management 218*  
*Architectural Issues - Integrating the Health-care  
Enterprise 223*  
*Architecture - Beyond DICOM 226*  
*Electronic Medical Record 227*  
*Summary 229*

**CHAPTER 9**

**Templates: Bringing order to chaos  
... 231**

*What is a Template? 231*  
*Other Template Features 235*  
*Templates and the Meaning of Codes 236*  
*Template Authoring 236*  
*Templates and Presentation 237*  
*Template Proposals for Common Patterns 238*  
*Diagnostic Imaging Report Outline 238*  
*Classification of Procedures 239*  
*Descriptions of Common Findings 240*  
*Measurement Templates 242*  
*Normal Ranges for Measurements 246*  
*Negation Templates 246*  
*Identifying Which Templates Are In Use 247*  
*Summary 249*

---

---

**CHAPTER 10**

**Internationalization  
and Localization 251**

*Basic Text Encoding: Single Byte Character  
Sets 252*

*Mixing and Matching: Code Extension  
Techniques 256*

*Multibyte Character Sets 260*

*Why not Unicode? 263*

*Character Sets and Conformance 264*

*Internationalization of Codes 265*

*Concept Modifiers and Languages 267*

*Identifying Languages 268*

*Designating Language of Names and Values 270*

*Equivalent Meanings 272*

*Reality Check - Codes and Languages in  
Practice 274*

*Summary 274*

**CHAPTER 11**

**Security: Privacy and  
Authenticity 277**

*What is "Security?" 277*

*DICOM and Security 278*

*TLS/SSL 279*

*TLS and Authentication of Individual Identity 280*

*TLS, Confidentiality and Integrity 283*

*Security Features Enabled by DICOM and TLS 283*

*Digital Signatures 284*

*Digital Certificates 285*

*Digital Signatures in DICOM 286*

*Validating Signatures and Bit Preservation 288*

*DICOM Media Security 289*

*Legal and Regulatory Issues 291*

*Summary 292*

**CHAPTER 12**

**Other Standards: HL7, MIME, XML,  
COAS 295**

*HL7 101 296*

*Encapsulation in HL7 Messages 298*

*MIME Encapsulation 299*

*E-mail 301*

*HTTP, MIME and DICOM 303*

*Transcoding and HL7 303*

---

---

<i>HL7 Clinical Document Architecture</i>	305
<i>XML: Beyond the Hype</i>	306
<i>What is XML?</i>	307
<i>Alternative XML Encoding Issues</i>	309
<i>A Standard XML SR Encoding?</i>	310
<i>XML and Presentation</i>	312
<i>HL7 CDA: Structure and Use of XML</i>	313
<i>CDA Coded Entries</i>	315
<i>Transcoding CDA and SR</i>	316
<i>Clinical Observations Access Service</i>	320
<i>Summary</i>	323

## **CHAPTER 13**

### **Trees, Traversal and Transformation 325**

<i>Tree Data Structures</i>	325
<i>Object Model</i>	326
<i>Document Object Model</i>	328
<i>SR Object Model</i>	329
<i>Using an SR Object Model</i>	333
<i>Iterators with Names</i>	336
<i>An Alternative Approach - Event Streams</i>	337
<i>Transformation</i>	339
<i>XSL-T</i>	340
<i>Using XSL-T with DICOM SR</i>	341
<i>Summary</i>	342

## **ANNEX A**

### **Walking Through the Standard 345**

<i>Information Object Definitions</i>	345
<i>SOP Classes</i>	347
<i>Data Dictionary</i>	347
<i>Encoding</i>	348
<i>Files and Media</i>	348
<i>Network</i>	349
<i>Conformance</i>	349

## **ANNEX B**

### **Basic Encoding Principles 351**

<i>Encoding</i>	351
<i>Transfer Syntaxes</i>	353

---

---

<i>ANNEX C</i>	<i>List of Codes</i>	<i>355</i>
<i>ANNEX D</i>	<i>List of Attributes</i>	<i>363</i>
	<i>Attributes Sorted by Tag</i>	<i>364</i>
	<i>Attributes Sorted by Name</i>	<i>367</i>
<i>ANNEX E</i>	<i>Structured Reporting Object Model</i>	<i>373</i>
	<i>Java</i>	<i>373</i>
<i>Index</i>		<i>379</i>



---

---

<i>Summary of Basic Code Sequence Macro</i>	<i>41</i>
<i>Coding Schemes listed in HL7, ASTM and/or DICOM</i>	<i>46</i>
<i>Summary of Enhanced Code Sequence Macro</i>	<i>61</i>
<i>Waveform IODs and Storage SOP Classes</i>	<i>148</i>
<i>Structured Reporting IODs and SOP Classes</i>	<i>163</i>
<i>Summary of SR SOP Class Specific Constraints</i>	<i>164</i>
<i>Mammography image module extract</i>	<i>232</i>
<i>Hypothetical template for mass characteristics</i>	<i>233</i>
<i>Hypothetical template that invokes mass characteristics as siblings</i>	<i>234</i>
<i>Hypothetical template that invokes mass characteristics as children</i>	<i>235</i>
<i>Hypothetical template with multiple nesting levels</i>	<i>235</i>
<i>Measurement units for distance, area and volume</i>	<i>245</i>
<i>Summary of Template Identification Macro</i>	<i>248</i>
<i>Sources of Single Byte Character Sets (with GO ISO 646)</i>	<i>254</i>
<i>Correspondence between Languages and Latin Alphabets</i>	<i>255</i>
<i>Single Byte Character Sets with Code Extensions</i>	<i>259</i>
<i>Multibyte Character Sets with Code Extensions</i>	<i>260</i>
<i>SR Object Model - Interfaces</i>	<i>333</i>
<i>SR Object Model - Methods for Interface Node</i>	<i>335</i>
<i>Value Representations</i>	<i>352</i>
<i>List of Codes Used</i>	<i>355</i>
<i>List of Attributes used in Structured Reporting, Sorted by Tag.</i>	<i>364</i>
<i>List of Attributes used in Structured Reporting, Sorted by Name.</i>	<i>367</i>



FIGURE 1.	Simple example of a DICOM Structured Report	30
FIGURE 2.	An SR Tree with References (Directed Acyclic Graph)	33
FIGURE 3.	Composite Objects Represent Multiple Entities	37
FIGURE 4.	Humans see headings, computer scientists see trees ...	68
FIGURE 5.	Reference Value Types	84
FIGURE 6.	Spatial Coordinates With Sub-Pixel Resolution	94
FIGURE 7.	Spatial Coordinates for Distance Along a Line	95
FIGURE 8.	Spatial Coordinates for Closed Objects	96
FIGURE 9.	Spatial Coordinates for Angles Between Lines	96
FIGURE 10.	Spatial Coordinates for Rectangles Including Rotation	97
FIGURE 11.	Temporal Range Types for Temporal Coordinates	99
FIGURE 12.	Purpose of Reference for Coordinates and Images	102
FIGURE 13.	Example of a Simple SR Tree with One Level of Children	107
FIGURE 14.	Implicit Numbering of Nodes in the SR Tree	108
FIGURE 15.	Example of Nesting of Document Macros	109
FIGURE 16.	Diagram Style to Illustrate SR Trees - Vertical	111
FIGURE 17.	Diagram Style to Illustrate SR Trees - Horizontal	111
FIGURE 18.	Diagram Style to Illustrate SR Trees - E-R Form	112
FIGURE 19.	Graphical Example of a Text SR with Headings	120
FIGURE 20.	Sharing Has Properties and Inferred From By-Reference	125
FIGURE 21.	Implicit Numbering of Nodes for References	130
FIGURE 22.	Simple Image Reference Pattern ± Purpose	140
FIGURE 23.	Key Image Note Image Reference Pattern (IHE Y2)	140
FIGURE 24.	Effect of Contrast Transformations	142
FIGURE 25.	Spatial and Contrast Transformations	143
FIGURE 26.	Spatial Coordinates Selected From an Image	144
FIGURE 27.	Temporal and Spatial Coordinates with an Image	146
FIGURE 28.	Temporal and Spatial Coordinates	147
FIGURE 29.	Dictated Report Workflow	150
FIGURE 30.	The Role of Drawings	154
FIGURE 31.	Form Recognition and DICOM SR as an End Product	155
FIGURE 32.	Form Recognition and DICOM SR as an Intermediary	155
FIGURE 33.	"Library" of Image Acquisition Context	159
FIGURE 34.	Example of "Has Observation Context" Relationship	176
FIGURE 35.	Example of Inheritance of Observation Context	177
FIGURE 36.	Replacement of Observation Context	178
FIGURE 37.	Observation Subject for Obstetric Ultrasound	179
FIGURE 38.	Quoted Content and Quotation Mode	182
FIGURE 39.	Composite Objects Represent Multiple Entities	190
FIGURE 40.	Predecessor Documents Sequence	194
FIGURE 41.	Identical Documents Sequence	194
FIGURE 42.	Identical Documents with Predecessors	195
FIGURE 43.	Workflow Management Tasks	218
FIGURE 44.	Scheduled and Performed Procedure Steps	219
FIGURE 45.	Dependencies of Reporting Procedure Steps	220
FIGURE 46.	Reporting Step Decomposed into Smaller Steps	221
FIGURE 47.	IHE Transactions and Actors Related to Reports	224

---

---

<i>FIGURE 48.</i>	<i>Areas of a Single Byte Code (as per ISO 2022)</i>	<i>252</i>
<i>FIGURE 49.</i>	<i>Structure of Single Byte Codes (as per ISO 2022)</i>	<i>253</i>
<i>FIGURE 50.</i>	<i>TLS, Authentication and Access Control</i>	<i>282</i>
<i>FIGURE 51.</i>	<i>SR Object Model: Input Applications</i>	<i>330</i>
<i>FIGURE 52.</i>	<i>SR Object Model: Rendering Applications</i>	<i>331</i>
<i>FIGURE 53.</i>	<i>Potential Scenarios for Applying XSL-T</i>	<i>344</i>
<i>FIGURE 54.</i>	<i>Data Element Encoding</i>	<i>352</i>





---

# Preface

---

If you have picked up this book, then you already know that DICOM (Digital Imaging and Communications in Medicine) is the ubiquitous standard in the radiology and cardiology imaging industry for the exchange of images and image related information. You may know that DICOM is extending into other image related medical fields, such as visible light applications in pathology, endoscopy, dentistry, ophthalmology and dermatology. You are probably also aware that an image without its associated information is not very useful, and that management of these images and this information requires other workflow management services that DICOM addresses.

You may also have heard that DICOM is now venturing into territory where other standards have feared to tread, into the area of structured documents. In particular, it is within the scope of DICOM to develop standards for documents that incorporate references to images and associated data such as waveforms. The recently released DICOM Structured Reporting (SR) supplement does just that. This book endeavours to explain some of the principles behind DICOM SR, and to help you understand the standard itself.

## **Who This Book is For**

This book addresses how to encode structured content using the DICOM standard. Accordingly, the content is very technical. It is intended primarily for those people who have to make structured reporting work in a device or application. Structured reporting is a particular challenge since it brings together two relatively disconnected worlds: that of low level DICOM encoding and messaging, and that of high level report creation, storage, query and presentation applications.

The book is not an introduction to DICOM SR, nor is it an overview. The amount of detail may be overwhelming for those who are not implementers, although a brief annex on fundamental encoding principles is included. Interspersed throughout is

---

some material that is more general, architectural and philosophical. This may be of interest to physicians and administrators who must make purchasing decisions related to the integration of images, waveforms and reporting. Vendor marketing folks who need to select features from appropriate standards to support their customers' reporting needs may also find it useful.

Nothing written here should be construed as representing original creative or scientific work by the author. Rather, this book explains how to implement a standard that is largely the result of the work of others. Notably absent is any discussion of the academic basis for structured reporting. There is a growing body of scientific literature on the concepts behind structured reporting. No attempt has been made to exhaustively review the literature,<sup>1</sup> nor to discuss the pros and cons of different strategies. For example, there is little discussion of structured input alternatives, natural language parsing, or various approaches for generating readable output.

## Outline of This Book

*Chapter 1, Structured Reporting Concepts*, introduces the basic concepts of DICOM structured reporting. It describes what structured reporting is all about, how it fits in the context of the DICOM world, what is contained within a document, and how structured reports are managed.

*Chapter 2, Codes and Controlled Terminology*, examines the need for and use of coded entries in DICOM in general and SR in particular. It also describes how and where these codes are defined, organized and managed.

*Chapter 3, Content Items: Concept Names and Values*, discusses the individual content items that are used to construct an SR document. Containers, concepts and name-value pairs are introduced. How to encode various different value types is discussed in detail.

*Chapter 4, Encoding & Relationships: Building the Tree*, describes the building blocks of the structured report tree: the recursively nested encoding mechanism and explicit relationships. Discussion of what relationships are specified, what they mean, and how to use them, forms the bulk of this chapter.

*Chapter 5, Images, Waveforms, and Voice Audio*, examines the importance of incorporating references to images and waveforms in reports. In addition, the role of scanned documents, voice audio, speech recognition and forms are reviewed.

*Chapter 6, SOP Classes and Conformance*, discusses the choice of SOP classes available for implementing structured reporting, as well as issues that relate to conformance with the standard

*Chapter 7, Context: Who did what to whom, when...*, introduces the idea that it is important for a document to encode the context in which its content was created.

---

1. For those interested in reviewing the literature on structured reporting, a good start is the web site "<http://www.mcw.edu/midas/spider/references.html>".

---

Where context is defined, how it is encoded, how it defaults, and how it is inherited in a top-down traversal of the SR tree is addressed

*Chapter 8, Document Management: SR in the Real World*, describes how documents are managed such that they are stored and made available where needed. Composite object instances, unique identifiers, the document paradigm, copies and versions, states of completion and verification, query and retrieval and workflow management are discussed.

*Chapter 9, Templates: Bringing order to chaos ...*, discusses how templates define patterns of SR content by constraining the concept names, relationship types, value types and value sets that may be used for a particular application. How templates are defined and applied is described, together with various proposals for templates.

*Chapter 10, Internationalization and Localization*, examines the basic mechanisms of encoding text beyond the default character set. Alternative ways to handle localized meanings for codes are discussed, as are issues of multi-lingual representations of code meanings and text.

*Chapter 11, Security: Privacy and Authenticity*, describes mechanisms for protecting the confidentiality and integrity of structured reports in a distributed environment. General issues are discussed, as well as specific DICOM solutions to the problems of secure transmission, authentication of users and devices, access control and digital signatures.

*Chapter 12, Other Standards: HL7, MIME, XML, COAS*, examines how DICOM structured reporting may coexist with other medical informatics and Internet standards, In particular, the issues associated with integration with information systems and transcoding into other formats are discussed.

*Chapter 13, Trees, Traversal and Transformation*, is an implementation oriented chapter. It describes how structured reports may be internally represented during creation or parsing and introduces the concept of a standard interface for accessing opaque internal representations. Analogies are drawn with the world of XML tools. The use of style sheets and rule-based transformation languages for transcoding and rendering applications are considered.

*Annex A, Walking Through the Standard*, is provided for those who are not intimately familiar with the organization of the DICOM standard itself. Each major facility of the standard applicable to structured reporting is associated with references to the relevant parts and sections of the standard.

*Annex B, Basic Encoding Principles*, provides a brief introduction to how data elements in DICOM objects are encoded into a binary bit stream.

*Annex C, List of Codes*, summarizes all the coded concepts that are used in the examples in the book, both standard and private.

*Annex D, List of Attributes*, summarizes all the DICOM data elements that are required for implementing the structured reporting SOP classes, sorted by name and tag number.

---

*Annex E, Structured Reporting Object Model*, defines the Java language binding for the proposed Structured Reporting Object Model that was introduced in the chapter on “Trees, Traversal and Transformation.”

## **Versions of the Standard**

This book will make very little sense at a technical level unless read in conjunction with the standard. Great swaths of the standard are not repeated verbatim, since the intent is to explain what SR is about, not duplicate the specification.

The 2000 edition of the DICOM standard is used as a reference. This version already includes the final text of Supplement 23 (Structured Reporting), Supplement 30 (Waveform) and Supplement 33 (Grayscale Softcopy Presentation State), all of which are relatively recent additions. It does not include the proposed Supplement 53 (DICOM Content Mapping Resource), which is not yet sufficiently mature to be covered in anything but the broadest way.

The final drafts of the latest version of the standard and the final text of supplements and corrections are always available on-line at the NEMA web site “<http://medical.nema.org/dicom.html>”. However, the official standard is the printed standard, plus any subsequent final text supplements or corrections. Implementers should take care to use the printed documents as their ultimate reference.

In this book, when individual parts of the standard are referenced, the form used in the standard is adopted. That is, “PS 3.nn” refers to “part nn” (for example, “part 3” would be referenced as “PS 3.3”). References to specific sections within parts is avoided, since it is possible that sections in the standard may be renumbered in subsequent versions.

## **Conventions Used in this Book**

To make the book read more fluently and less like the standard itself, terms like “node” and “content item” are used interchangeably. Other terms like “leaf” and “terminal node,” “parent” and “ancestor,” “child” and “descendant” and so on are also used for variety. Capitalization or quotation of attribute and concepts names is largely avoided for readability. The word “attribute” is used generically, to mean “a property of something.” This is irrespective of whether the attribute is encoded as an entire SR content item, or as a single DICOM data element.

Sometimes a concept name will be written in lower case, as for example “observer.” At other times it might be written as “Observer,” or, when being really pedantic, as a tuple (209069, 99PMP, “Observer”). In a similarly inconsistent fashion, value types and relationships will sometimes be written in all capitals as they are actually encoded, such as CODE or HAS OBS CONTEXT, and at other times in lower case and expanded as “code” or “has observation context.” The goal is to avoid too many capitalized words which become obnoxious after a while (as any reader of the standard itself will attest to).

---

In order to compactly convey examples of structured report fragments without overwhelming the reader with encoding details, the shorthand of the form illustrated in the following example will often be used:<sup>2</sup>

```
<CONTAINER:"Findings">  
  <contains TEXT:"Finding"="enlarged heart">  
  <contains TEXT:"Finding"="widened mediastinum">
```

In this example, each content item is indicated between "<" and ">". The indentation implies nesting. A non-capitalized phrase indicates the relationship (e.g. contains, inferred from, has properties, etc.). A fully capitalized word followed by a colon indicates the value type (such as CONTAINER, TEXT, CODE, etc.). Following the value type is a name-value pair, with the name and value respectively enclosed in quotes and separated by an "=". The full tuple that represents a coded entry for a concept name or coded value is elided in most cases, and only the code meaning is shown.

When it is helpful to fully elucidate a coded entry, a tuple of the form:

```
(T-04000,SNM3,"Breast")
```

will be used, with the first entry indicating the code value, the second indicating the coding scheme designator, and the third the code meaning. The code meaning will always be enclosed in quotes. The other entries will only be enclosed in quotes if a comma occurs within the code value or coding scheme designator. The coding scheme version will usually not be included. Where space on a line is at a premium and the actual codes are not relevant, an abbreviated form is used that shows only the code meaning:

```
(,, "Breast")
```

Conveying the details of the DICOM encoding of fragments of SR objects also requires a convention, at least for those who tire of interpreting hexadecimal dumps of binary data. The general pattern that is used is to indicate the group and element number as a tuple of hexadecimal values, follow that with the attribute name (which does not appear in the actual binary encoding),<sup>3</sup> and then the value in quotes. For example:

```
(0008,0104) Code Meaning "Breast"
```

Where it is important, additional fields are added to indicate the value length and explicit value representation, though often these are elided. Sequences and sequence items are shown in their undefined length delimited form, as in:

```
(0040,a043) Concept Name Code Sequence  
(fffe,e000) Item  
...
```

---

2. This is not a shorthand described in the standard, but rather something created for the purpose of this book. It is not a novel idea. There have been various proposals for a "formal syntax" for describing SR documents in the past. The interested reader is referred to earlier drafts of Supplement 23 that describe such a syntax for expressing relationships.

3. Much to the chagrin of early detractors of ACR-NEMA and DICOM.

---

(fffe,e00d) Item Delimitation Item  
(fffe,e0dd) Sequence Delimitation Item

This should not be taken to imply that sequences and items may not be encoded with defined length (in which case the trailing delimitation items are elided in the binary encoding).

Where more extensive examples are used that involve nested sequence items, they may be shown using “>” signs to indicate the depth of nesting, with or without a numerical hierarchy (“1”, “1.1”, “1.2”, etc.) corresponding to the referenced content item identifier mechanism.

Since there is not yet consensus on which codes from which coding schemes will be used for common SR applications, a private coding scheme has been created for this book. Throughout the text it is referred to with a coding scheme designator of “99PMP”. The codes in this private scheme are summarized in the annex List of Codes. Whenever standard codes are available for concepts, they are used instead.

When encoded binary data is represented it will usually be shown as zero padded hexadecimal bytes, e.g. “0x0F”. Occasionally, zero-padded octal bytes will also be shown, particularly when discussing character encoding (since that is how it is done in the standard), e.g. “\017”. The Unix convention for special characters will also occasionally be used, such as ‘\n’ for new-line, ‘\r’ for carriage-return, as opposed to the “LF” and “CR” that are used in the standard.

The styles used for copy editing vary slightly from what is considered standard practice. Though in most cases terminal punctuation is included within quotations, in some cases this would be confusing. For example, if the contents within the quotation marks are a literal value defined by the standard, the trailing punctuation is placed outside the closing quote. Items in lists are not punctuated in a conventional fashion, as the most common style of capitalizing the first letter and adding trailing punctuation simply is unwieldy. References to book titles include an ISBN, which is not usually the case. This practice seems to make it easier to locate and order books on-line.

Extensive use of footnotes is made, rather than magazine style side bars or called-out text boxes. Footnotes turned out to be a convenient mechanism for adding details and incidental information that otherwise disrupted the flow of the body text. This may not be to every reader’s taste. Admittedly, the sheer volume of footnotes can sometimes be overwhelming.

## Acknowledgements

It has taken many years for Dean Bidgood to convince participants in DICOM that structured reporting is both feasible and practical. The breadth and depth of his vision, combined with the audacity of the challenge that SR presents to conventional assumptions about DICOM encoding, resulted in slow progress. Though Dean would be the first to point out that SR is the result of the contributions of many dedicated organizations and individuals, his own determination and perserverance in the face of adversity must stand as an example to us all. His graciousness in accepting less than constructive criticism is inspiring.

---

Contributors to the SR development effort include the College of American Pathologists (CAP) who hold the WG 8 (SR) secretariat and provide SNOMED, the National Electrical Manufacturers Association who hold the DICOM Committee secretariat, the American College of Radiology (ACR), who with NEMA began DICOM, the French Radiological Society (SFR), the American College of Cardiologists (ACC), the European Society of Cardiology (ESC), the American Academy of Ophthalmologists (AAO), the American Dental Association (ADA) and the American Society of Gastrointestinal Endoscopists (AGSE).

For reviewing the entire manuscript I am grateful to Janet Keyes, Julian Marshall, Richard Crane, Andrei Leontiev, Robert Dolin, John Carrino, Charles Parisot, Nicholas Brown, Cor Loef, Peter Kuzmak, Marco Eichelberg, Joerg Riesmeier and Dee Csipo, as well as Lawrence Tarbox, Lloyd Hildebrand, Steve Moore and Hidenori Shinoda for reviewing selected chapters. For many years of fruitful discussions about DICOM in general and SR in particular I would like to thank Dean of course, as well as the many other working group members who are too numerous to mention.

Thanks are also due to Greg Pisocky and Michael Ouslander from Adobe Government Systems Group for kindly supplying the copy of Framemaker used to prepare the manuscript, and Ken Lunde, also from Adobe, for assistance with Japanese fonts. Hidenori Shinoda assisted with the examples of Japanese names and characters and Seong-min Lim assisted with the Korean examples.

Finally, I would like to thank my wife Eleanor, for putting up with my obsession with things technological. Not to mention for foregoing my company on those occasions when I felt inspired to write rather than spend time in the city eating sushi and drinking sake in the company of our friends.

## **Feedback**

As always, I welcome comments, criticism and suggestions. These are best directed to my E-mail address at "mailto:dclunie@idt.net". For discussions with a broader audience, the Usenet newsgroup "news:comp.protocols.dicom" is a good forum.

David A. Clunie

Bangor, Pennsylvania

October 20th, 2000.



# ***Structured Reporting Concepts***

---

In this chapter, the basic concepts of structured reporting will be presented, including:

- ▶ what structured reporting in DICOM is all about
- ▶ how it fits in the context of the DICOM world
- ▶ what is contained within a DICOM SR document
- ▶ how reports are managed

The term “structured reporting” means different things to different people. A radiologist, thinking in terms of conventional reports, may envisage a document that consists of a hierarchy of headings containing blocks of text, perhaps with a few codes or keywords at the end to summarize the findings. The author of a software package for making obstetric ultrasound measurements may visualize a nested hierarchy of related numeric measurements (like femur length, each identified with individual codes, then aggregated to provide more general measurements, such as averages or estimates of fetal age). The manufacturer of a cardiac catheter lab information system may see a mechanism for encoding successive time-stamped procedure log entries, consolidated from different equipment sources.

From the perspective of DICOM Structured Reporting (SR), what unifies these different views are:

- ▶ the presence of lists and hierarchical relationships
- ▶ the use of coded or numeric content in addition to plain text
- ▶ the use of relationships between concepts
- ▶ the presence of embedded references to images and similar objects

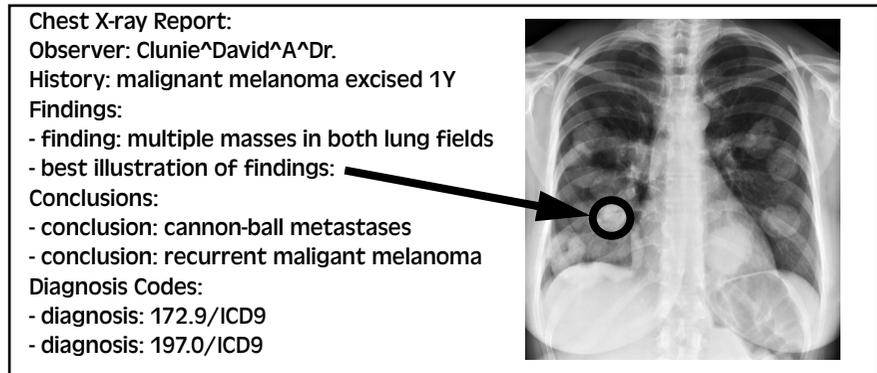


FIGURE 1. Simple example of a DICOM Structured Report

---

## Not Just Reports ...

Figure 1 illustrates in a stylized form how DICOM SR may be used for traditional reporting.

However, a structured report (or in a more general sense, a “structured document”) is defined more by how it is constructed than by what it contains. The word “report” in the name is really a misnomer, since a DICOM SR can convey any kind of structured content, not just reports. SR documents can be used wherever there is a need for lists or hierarchically structured content, or a need for coded concepts or numeric values, or a need for references to images, waveforms or other composite objects.<sup>1</sup>

SR objects need not be complex documents, nor always intended for direct rendering into a human-readable form. Small objects that contain little more than an image reference and an explanatory note are useful for flagging images of significance (a “key image” note). The ability to “collect” references to different types of object into a single persistent document suggests interesting possibilities.

## Semantics not presentation

In DICOM SR, each document encodes only what is meant, not how it is intended to be displayed, printed or otherwise presented.<sup>2</sup> The meaning is required to be unambiguous. It cannot be dependant on assumptions about how it might be presented.

- 
1. A “composite” object in DICOM is one that contains attributes of multiple entities, such as patient and study, as well as individual document information.
  2. There is one exception to this that allows for the inclusion of references to presentation states for grayscale images. These do not apply to presentation of the report itself, however.

In this sense, DICOM SR may be compared to XML (eXtensible Markup Language)<sup>3</sup> as opposed to HTML (HyperText Markup Language).<sup>4</sup> Whereas HTML conveys explicit presentation information without meaning, XML contains tags with meaning but without explicit or implied presentation. Just as it may be necessary to combine an XML document with some form of presentation tool such as a style sheet written in XSL (eXtensible Stylesheet Language)<sup>5</sup> or CSS (Cascading Style Sheets)<sup>6</sup> in order to render it, so it may be necessary to use an application that understands how to turn DICOM SR content into an appropriate form.<sup>7</sup>

## Constraints, IODs and Templates

Carrying the XML analogy further, just as the basic encoding of a “well-formed” XML document is defined by the XML Recommendation, so the encoding of a DICOM SR object is defined by the DICOM Standard. The low level encoding is specified in PS 3.5. The basic framework of the SR structure is contained in the definition of the SR information objects in PS 3.3. Whether or not an XML document contains information according to a pre-defined structure and makes use of appropriate tags (i.e. is “valid”) is defined in its DTD (Document Type Definition), if present.

The SR information object definitions (IODs) in PS 3.3 serve a similar function to an XML DTD, at least to the extent that valid combinations of atomic components like value types and relationships are specified.<sup>8</sup> However, an XML DTD, and other more advanced XML structuring tools like XML Schema, are really intended to constrain a generic mechanism to a specific domain. From this perspective, the DICOM SR contents may be constrained further by using the “templates” that will in future be defined in DICOM PS 3.16.<sup>9</sup>

Templates may be used to achieve consistency without being mandatory, when used with one of the general purpose SR SOP classes. Such is usually the case for general purpose reporting. In other cases, a specific template may be linked to a specific SOP class. This constrains flexibility in order to achieve greater interoperability. For example, the proposed Mammography CAD (Computer Assisted Detection) SR SOP class will have a highly specific structure that is defined by templates rather than the IOD.<sup>10</sup>

---

3. See “<http://www.w3.org/XML/>”.

4. See “<http://www.w3.org/MarkUp/>”.

5. See “<http://www.w3.org/Style/XSL/>”.

6. See “<http://www.w3.org/Style/CSS/>”.

7. *At present, there is no style sheet mechanism that is specific to DICOM defined.*

8. *Each DICOM composite IOD, when paired with the Storage Service Class, results in a Service-Object Pair (SOP) Class, which is the level at which conformance is defined and negotiation is performed.*

9. *See Supplement 53.*

10. *See Supplement 50.*

## Content

The interesting part of a DICOM SR document is its “content,” that is the headings and text and other types of information that convey the “meaning” of the document. For example, a document might contain content that describes a mass:

```
CODE: "Finding" = "Mass"  
NUM:  "Diameter" = "1.3" "cm"  
CODE: "Shape"    = "Round"  
CODE: "Margin"   = "Well-defined"
```

In a DICOM SR document all information is conveyed by individual “content items.” Each content item is a “name-value pair.”<sup>11</sup> The “name” is referred to more precisely as the “concept name.” It is always defined by a code rather than by free text. This facilitates indexing and searching. Concept names are also used to define headings of containers, and to indicate the purpose of references to images and waveforms.

Each content item can also have a value. The value may be of various types, including:

- ▶ plain text
- ▶ coded values
- ▶ numeric values (with units)
- ▶ person names, dates and times
- ▶ references to DICOM images, waveforms and other composite objects
- ▶ spatial and temporal coordinates in referenced objects

The value type is always explicitly specified and may be constrained to a limited set of types by the SOP class of the object. Templates may further constrain the range of possible value types. For example, when a physician needs to be identified for some purpose, a person name or coded value type might be specified.<sup>12</sup>

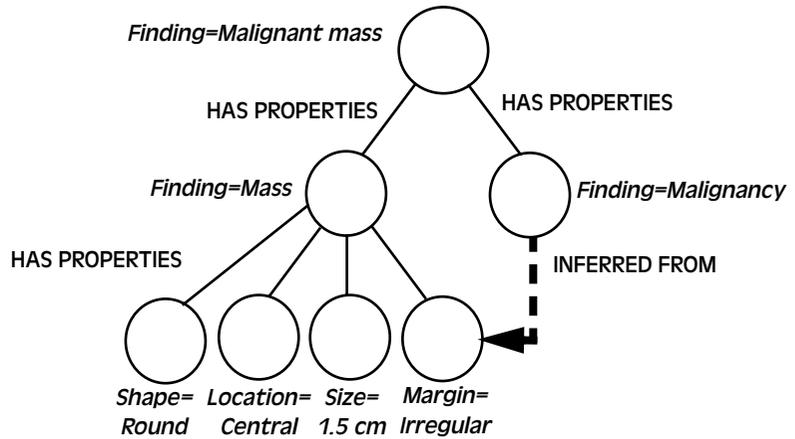
Content may be multi-lingual, localized and internationalized by using the DICOM specific character set mechanism. The names and values of concepts may be conveyed by codes which are not specific to any particular country or region. Alternative meanings in different languages for the same concept name may also be explicitly conveyed.

References to external objects such as images and waveforms are constrained to be to DICOM objects only. That is, one can reference an instance of a DICOM CT Image Storage SOP class, but not an arbitrary GIF or TIFF file. Similarly, one can reference a DICOM Basic Voice Audio Waveform Storage SOP class instance, but not an AIFF or WAV file. This decision was made in order to constrain SR to a level of interoperability that may be specified within the scope of DICOM by the conformance statement. An SR document can be decoded and rendered without requiring a plethora of unconstrained proprietary plug-ins.

---

11. This is true of all content items other than containers, not just leaves of the tree.

12. In this sense, template definitions are analogous to the Module tables of traditional DICOM composite IODs. DICOM SR differs from other IODs in that modules provide only a general framework, and templates are used to describe domain specific restrictions.



**FIGURE 2. An SR Tree with References (Directed Acyclic Graph)**

One cannot reference or include pre-formatted text, such as HTML, Word, Postscript or PDF. Only unformatted text may be included, though embedded new lines are permitted. This restriction further separates presentation from semantics.

The ability to specify value types is a key DICOM SR feature that distinguishes it from other structured document formats that only allow for plain character text.

## Trees and Relationships

Content items are not much use unless they are linked together in some manner. In DICOM SR, all content items are encoded as a single tree with a single “root” content item. The root content item conveys the “document title.” Its children and their descendants convey the content of the document. An SR may be as simple as a “flat” list of content items below the document title. Alternatively, an SR may be as complex as a directed acyclic graph (DAG). In a DAG, even though content is encoded as a tree, individual content items may reference content items other than their parents or children.<sup>13</sup> This is illustrated in the example shown in Figure 2.

References to internal nodes within the tree of another SR document are not allowed. The content tree must be fully contained within a single SR document. When it is necessary to use part of the content of another document, it must be replicated in place (“quoted”). This is a consequence of the requirement that each document be a stand-alone entity. When objects are referenced, they are referenced as a whole. The single exception is for references to objects which contain bulk data, such as images and waveforms, regions of which may be selected by coordinates.<sup>14</sup>

13. The two simpler SOP classes (Basic Text SR and Enhanced SR) forbid the use of this reference mechanism for simplicity’s sake, and are thus constrained to be strictly trees.

A hierarchical structure is useful to organize content items, but it is not sufficient. It is also necessary to specify “relationships” between content items. This is achieved by requiring every content item to have an explicit relationship with its immediate ancestor. Put another way, every content item has an explicit relationship with every immediate descendant. The relationship may be different for each descendant. The types of relationships that are defined include:

- ▶ simple containment, which implies no semantics, other than grouping and ordering of descendants
- ▶ description of properties of the parent
- ▶ description of the source of the inference or conclusion stated in the parent
- ▶ description of the image or waveform content from which the parent coordinates were selected
- ▶ description of the context in which an observation or acquisition was made
- ▶ modifiers of the concept of the parent

Relationships may be specified with children “by value,” or “by reference” to content items elsewhere in the tree.

Explicit relationships are one of the features that distinguishes SR from other structured document formats like XML. Other formats provide the ability to hierarchically nest content, but there is no definition of what it actually “means” to be nested inside other content. In SR, one can explicitly specify relationships between parents and children. For example:

- ▶ that a list of features (such as diameter and margin) are *properties* of a content item (such as a mass)
- ▶ that a conclusion (such as malignancy) was *inferred from* a feature (such as the presence of a cluster of pleomorphic microcalcifications)

Without explicitly stated relationships, in other formats the user is left to make assumptions what semantics are implied by the nesting from the names of concepts.

## Coded Concepts and Values

Simply organizing plain text into a structured document provides only limited benefit. Effective searching and matching also requires consistency in the use of terms for concepts and values. Such consistency is achieved through the use of “controlled terminology” chosen from dictionaries (lexicons) of coded concepts. DICOM uses “coded entries” to embed such terms. A coded entry in DICOM consists of:<sup>15</sup>

---

14. *It is not practical to convey entire images and waveforms inside the report itself. This would lead to reports that were enormous in size.*

15. *There are other optional extended fields for providing version information, indicating that a code is a private extension of a standard scheme, conveying the list of terms from which the particular code was chosen, etc. These may be ignored for most purposes. Further detail is provided in the chapter on “Codes and Controlled Terminology.”*

- ▶ a “code value” that specifies the term
- ▶ a “coding scheme designator” that specifies the dictionary that defines the term
- ▶ a “code meaning” that contains a human readable description of the term

For example, when one wishes to refer to the breast (say, as the anatomical location of a mammographic examination) one can use the triplet (T-04000,SNM3,“Breast”). Any application can then search for the combination of T-04000 and SNM3 to unambiguously locate content referring to this concept, regardless of punctuation, spacing, padding, character set or language choice. Note that it is the combination of code value and coding scheme designator that uniquely defines the concept referred to by the coded entry. The code meaning field is purely for convenience of presentation by applications that do not possess a lexicon in which to lookup the code.<sup>16</sup> One should not match during a search or make decisions predicated on the code meaning field.

The codes used commonly in DICOM will in future be defined in PS 3.16, and are mostly drawn from widely-used coding schemes like SNOMED<sup>17</sup> and LOINC<sup>18</sup> (when necessary, with the permission of the appropriate organizations). No royalties will be required to be paid by the users of the codes that will be listed in PS 3.16.

The concept names in an SR document are always coded entries. Though the value may be plain text when necessary, it may also be a coded entry. The units of numeric values are always coded entries, as is the purpose of reference to an image, waveform or coordinate.

Sometimes a concept name is insufficiently specific, needs further description or qualification, or needs explicitly encoded multiple language meanings. In such cases, a content item may be decorated with children that have a “has concept modifier” relationship with their parent. The concept name can thus be “modified” by further coded or plain text information. For example, a document title (root node) that is the coded equivalent of “chest x-ray report” may be modified by a coded entry that indicates the “views”=“pa and lateral”, rather than using a single complicated code indicating “chest x-ray, pa and lateral views, report”. This is referred to as “post-coordination” rather than “pre-coordination” of codes. The use of modifiers allows one to define and search for generic terms, then refine a search with more granularity. It also avoids a combinatorial expansion of pre-coordinated codes.

## **Living Without Codes**

For those who are intimidated by codes, structured reporting still provides a useful framework. The only place where coded entries are absolutely required is for the concept names, and a small subset of codes that are fairly generic will suffice in many cases.

---

16. Such as may be the case when private or local coding schemes are used.

17. Systematized NOMenclature of human and veterinary MEDicine, College of American Pathologists.

18. Logical Observation Identifier Names and Codes

For example a small subset of codes meaning “radiology report,” “procedure,” “findings” and “conclusion” might be sufficient to encode a plain text structured report. Extensive use of plain text is still required for many real-world applications. It is not always easy to obtain “structured input” without significant effort on the part of the user. Plain text transcribed from voice dictation remains the state of the art for many reporting applications.<sup>19</sup>

Implementers should not be frightened by the coded entry facilities of structured reporting. The framework can be used in a simplistic manner to useful effect. In the degenerate case, a single blob of unformatted text with only a title can be embedded in a DICOM structured report, even though it lacks structure. Since all DICOM structured reports are composite objects (just like images), they can subsequently be stored and managed in a conventional DICOM archive, without the need for special services. Equally simplistic approaches can be used to create objects that identify “key images” selected from a larger study, without the need for extensive codes or descriptive information.<sup>20</sup>

The spectrum of generic SR SOP classes ranges from basic text through enhanced to comprehensive. This allows implementers of simple reports to avoid advanced features, yet still provides a meaningful compliance mechanism.

## Report Management

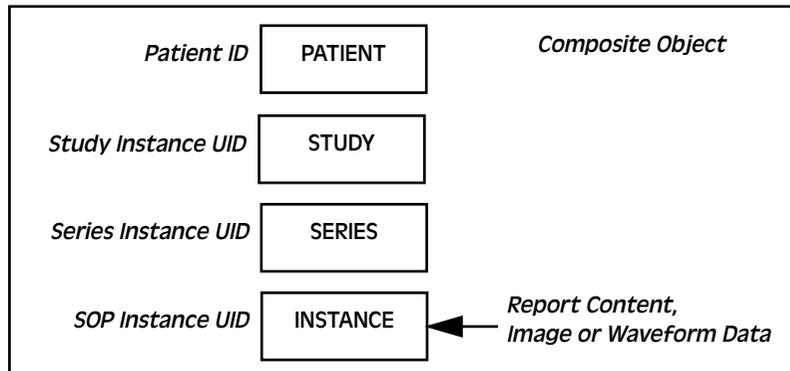
Structured reporting requires more than creating, encoding and storing content. Once created, documents must be managed so that they are available where and when they are needed. One needs to define where documents come from, when to create them, where documents are sent, what version of a document is in use, what the current state of a document is, and how to locate relevant documents. The SR IOD defines a module separate from the content tree that contain this information (the SR Document General Module).

Structured reports are DICOM composite instances just like images and waveforms. As such, they contain attributes to identify and describe the entities in the composite information model. This includes information about the patient, study, study component, series and instance (see Figure 3). Since SR documents are instances of composite SOP classes they are assigned unique identifiers (UIDs). DICOM composite instances are normally persistent beyond the scope of their transmission; they are “documents” rather than “messages.” In a message-oriented paradigm, they can be transient and be discarded after they have been used.

---

19. Most physicians, especially radiologists, simply will not type their own reports. Continuous speech recognition systems work for some applications, especially those with highly repetitive and stylized input. A speech recognition system that requires training and regular feedback on performance may be tolerated by regular users like diagnostic radiologists. It may not work as well for other physicians whose contact with the system is sporadic.

20. For example, this “key image note” is one of the patterns specified for the Integrating the Health care Enterprise (IHE) demonstration in Year 2.



**FIGURE 3. Composite Objects Represent Multiple Entities**

The content and management information encoded in a structured report cannot be altered without creating a new instance. Amendments or revisions are required to be new document instances. Specific rules govern the references between duplicates and versions of a structured report. In particular, all identical documents and prior versions of a document are referenced by the most recent document.

DICOM structured reports are also encoded with status about their completion and verification. This embedded status information is only intended to be sufficient to identify the prevailing, most complete, verified report. It may not be sufficient to manage the fine granularity necessary to create, edit and distribute reports. A verified report is one that has been “attested to” (“signed”) by one or more individuals who take responsibility for its content.

SR documents also contain lists of other DICOM composite instances that are either the subject of the current procedure or other information relevant to the content of the document. These lists can be used to “pre-fetch” instances that are necessary to render the document. They may also be used to record what information was available to the creator of the document.<sup>21</sup> In some applications these lists may be empty.

The DICOM composite instance Query-Retrieve (QR) service can be used to find structured reports as well as other instances. There are some limitations that are a consequence of structured reports making extensive use of nested sequences. However, at the very least, the title of the document should be easily retrievable. The QR service defines a baseline “hierarchical” model and an optional “relational” model. The hierarchical model requires knowledge of what study and series<sup>22</sup> a report is contained in before an instance of a report can be located or retrieved. Repeated, successively refined, queries may be required with the hierarchical model. It is hoped that

21. Regardless of whether or not they actually made use of all the information that was available.

22. All composite objects are grouped into series. A series can only contain objects of the same modality, generated by the same equipment. In the context of SR, a series has little meaning, except perhaps that reports, images and waveforms cannot be in the same series.

deployment of structured reporting will stimulate broader support for the relational query model. This would make structured reports and referenced objects easier to locate.

It is also desirable to manage the workflow for creating and distributing reports. The standard is currently being extended to provide interpretation worklists similar to the existing modality worklist and performed procedure step SOP classes.<sup>23</sup> This facility will provide worklists of scheduled steps that describe what tasks are to be performed and what information is needed as “input,” as well as a mechanism for signaling what steps have been performed and what “output” has been generated.

The design of an entire architecture to support reporting applications is beyond the scope of the standard. DICOM is only a standard for communication across boundaries between devices. It is likely, however, that structured reporting will play a small but key role in any complete electronic record of the future, serving to standardize sources of images, waveforms and related reports.

## Summary

This chapter has provided a high level overview of some of the main structured reporting concepts, each of which will be revisited in greater detail in subsequent chapters. Some of the key features of DICOM structured reports are that they:

- ▶ can be structured documents of any kind, not just traditional radiology reports
- ▶ can contain lists and hierarchical relationships
- ▶ may contain embedded references to images and similar objects
- ▶ encode only semantic information and not presentation information
- ▶ are constrained in their form by general purpose SOP classes and more specific templates
- ▶ contain content items that consist of name-value pairs, with codes for names and a variety of value types
- ▶ make use of explicit relationships between content items, rather than depending on implicit containment
- ▶ allow for structures more complex than trees through the use of “by-reference” relationships using content item references
- ▶ make extensive use of coded or numeric content in addition to plain text
- ▶ may make use of controlled terminology to facilitate automated understanding of content, searching and internationalization
- ▶ can be used with a minimal subset of codes to define document titles and major headings when necessary
- ▶ can be managed using conventional DICOM services like storage, query and retrieval, storage commitment, worklist and performed procedure step

---

23. Supplement 52, Interpretation Worklist.

# ***Codes and Controlled Terminology***

---

One of the fundamental requirements of a structured documentation and retrieval system is the ability to use controlled terminology (coded entries) rather than plain text. Natural language is notorious for its subtlety and ambiguity, and there is little room for either in medical reporting applications. The use of precisely defined coded terms rather than plain text to describe concepts and values also enables more precise searches. Synonyms can be unified with a single coded entry for a concept, or by a mapping between different codes for synonyms. Rendering into a locally preferred language can be made independent of the meaning and encoding of the term itself.

The use of codes is not novel. Codes have been used since the beginning of computing, primarily for compactness. In DICOM, coded entries have already been used in some image objects to replace strings for defined terms and enumerated values.

This chapter will examine the use of coded entries in DICOM in general and SR in particular, as well as discuss how and where these codes are defined, organized and managed. Also considered are some historical oddities that have crept into the standard as the use of controlled terminology has matured.<sup>1</sup>

## **Basic Concepts of Coded Entries**

To precisely define a coded entry, one needs:

- ▶ an encoded value for the code, the “Code Value”
- ▶ a designator of the scheme that defines the meaning of the code, the “Coding Scheme Designator”

---

1. Such as the use of the 99SDM coding scheme designator in some enumerated context groups, and references to the now superseded SNOMED-DICOM Microglossary (SDM).

The code value is very likely to be compact, rather than human readable, so in order to be able to display or otherwise render it one may also need:

- ▶ a human readable text meaning, the “Code Meaning”<sup>2</sup>

For example, in the DICOM digital mammography image object, the anatomical region is designated by a coded entry that describes the breast, specifically:

- ▶ a Code Value of “T-04000”
- ▶ a Coding Scheme Designator of “SNM3”
- ▶ a Code Meaning of “Breast”

As promised, the code value itself is a meaningless, unreadable string. The presence of a code meaning in the encoded data stream allows a naive application to simply display the meaning of the code. Note however, that the actual specification of the concept being coded is by the unique combination of code value and coding scheme designator, irrespective of the transmitted code meaning. To be pedantic about this:

- ▶ the code value by itself is not sufficient to uniquely specify the concept - two different concepts could share the same code value if the coding scheme designator were different
- ▶ the actual string that is encoded as the code meaning may vary despite having the same code value and coding scheme designator, for example as a consequence of there being synonyms for the same concept unified by the same code<sup>3</sup>
- ▶ as a consequence of the foregoing, one should never make conditional decisions based on the value of code meaning, nor on the code value alone; one must always test the combination of code value *and* coding scheme designator<sup>4</sup>

In this example, the coding scheme was designated as “SNM3” which corresponds to SNOMED Version 3. Sources of codes will be discussed later in this chapter.

## DICOM Encoding

The pattern of describing three parameters of a coded entry is used so often it is referred to informally as a “triplet code.”<sup>5</sup> These triplets are always encoded in DICOM as an item of a “code sequence” attribute.<sup>6</sup> There may be more than one item if there is more than one coded “value.” Essentially, the simple string value of a conventional attribute is replaced by a sequence of items, each of which contains the three attributes that specify the coded entry. For example, the old View Position

---

2. This field is referred to as “display name” in HL7.

3. There is also the possibility that the meaning may be localized with different language translations.

4. For example, ‘if (CodeValue==“T-04000” && CodingSchemeDesignator==“SNM3”)’.

5. The term “triplet code” originally comes from ASTM E1238.

6. Though not always as the same attribute. For example there is an “Anatomic Region Code Sequence”, a “Procedure Code Sequence”, etc. An attribute that has a coded sequence value (almost) always has the words “Code Sequence” in the name.

attribute used in early projection radiographic images in DICOM, is replaced in later objects by the View Code Sequence attribute. Instead of:

(0018,5101) View Position "AP"

one now has:

(0054,0220) View Code Sequence  
 (ffff,e000) Item  
 (0008,0100) Code Value "R-10206"  
 (0008,0102) Coding Scheme Designator "SNM3"  
 (0008,0104) Code Meaning "antero-posterior"  
 (ffff,e00d) Item Delimitation Item  
 (ffff,e0dd) Sequence Delimitation Item

In the standard, the use of a code sequence is specified by invoking the "Code Sequence Macro." The invocation of the macro may specify constraints on which codes may be used (the so-called "Value Set") in the form of baseline, defined or enumerated "Context Groups." These will be discussed later. The basic set of attributes used in the macro, together with their requirement types,<sup>7</sup> are shown in Table 1.

TABLE 1. Summary of Basic Code Sequence Macro

Attribute	Tag	Type
Code Value	(0008,0100)	1
Coding Scheme Designator	(0008,0102)	1
Code Meaning	(0008,0104)	1
Coding Scheme Version	(0008,0103)	1C

Notice that when the code sequence macro is used, all three attributes of the triplet code are mandatory. Specifically, one always has to send the code meaning. This was not always the case in DICOM. The requirement for code meaning was changed from optional to mandatory in order to be sure that receiving applications which were not in possession of the "dictionary" or "lexicon" (in which to look up the meaning of a code) could use the supplied meaning when a human readable form was needed.<sup>8</sup> Indeed, in some other standards like HL7 and ASTM, the code meaning ("display name" or "text name/description") is not always transmitted.<sup>9</sup> The bottom line is that the receiving application really should have an appropriate lexicon, and where possible should use the meaning from the lexicon, falling back on the supplied code meaning only if the lookup fails.

7. The "type" field in a composite object definition specifies whether or not an attribute is required (1), must be present but may be zero length if "unknown" (2), or is optional (3). In addition, a type 1 or 2 attribute's presence may be conditional (1C or 2C).

8. This change was made in Supplement 36, Codes and Controlled Terminology, which was adopted as part of the standard on September 1, 1998.

9. In HL7 and ASTM E1238, coded entries are conveyed in the CE (Coded Element) data type.

For historical reasons,<sup>10</sup> in PS 3.4, some of the older normalized services still specify code meaning as optional rather than mandatory. Though most implementations send code meaning, one cannot depend on its presence. This applies, for example, to the modality worklist and modality performed procedure step SOP classes. The same is also true of a few code sequence attributes in the nuclear medicine image IOD (in PS 3.3).

There are other (optional) attributes defined in the code sequence macro which are not yet in widespread use. These will be discussed in a later section.

## **Coding Scheme Designator and Version**

What about Coding Scheme Version, which is also potentially mandatory, but hasn't been mentioned in the "triplet code" examples? The condition on the coding scheme version attribute is that it is "required if the value of Coding Scheme Designator is not sufficient to identify the Code Value unambiguously." In many cases the coding scheme designator is sufficient, hence the version does not need to be sent. For example, a coding scheme designator of "SNM3" is regarded as sufficient to identify SNOMED Version 3. Codes derived from subsequent versions of SNOMED (SNOMED-RT) are designated in DICOM only by "SRT" which does not identify the version, hence an explicit coding scheme version is necessary.

Is it necessary to examine the value in coding scheme version when looking up a code in a lexicon? The answer to this is not entirely clear, but is probably "yes." Certainly, the version of a coding scheme may change as new codes are added. If one used an older dictionary, the meaning of a newer code might not be found. Alternatively, older codes might have been removed. In either case, the transmitted value of coding scheme version could be ignored. A problem arises, however, if the meaning of a code is actually changed in subsequent versions. Whether or not any of the coding schemes would ever do this is unclear. Still, to be sure, it may be safest to always check both coding scheme designator *and* coding scheme version (if present) when looking up code values in dictionaries.

A historical peculiarity is the "99SDM" coding scheme designator used in some of the older objects.<sup>11</sup> Originally the idea was that the coding scheme designator field would identify not the owner of the codes, but the mapping of those codes used by DICOM. This is a so-called "mapping resource," which was named at the time the "SNOMED-DICOM Microglossary" (SDM). All the coding scheme designators that begin with "99" are defined to be private coding schemes.<sup>12</sup> The designation "99SDM" was ini-

---

10. Specifically, for fear of breaking existing implementations.

11. One will still encounter "99SDM" in nuclear medicine images, where it was used as an enumerated value. One may even encounter older x-ray angiography images with a coding scheme designator of "99DEV". This was used in the letter ballot text draft of Supplement 4, but was changed in the final text.

12. The definition of private "99" coding schemes comes from HL7 and was formerly replicated in Annex D of PS 3.3 of DICOM.

tially used with the intent that a “non-local” coding scheme designator would be assigned later.

This confusing situation has now been rationalized such that the coding scheme designator really does refer to the authority responsible for the codes themselves. Hence “SNM3” is used instead of “99SDM”. The mapping resource can be specified with an additional explicit attribute, as will be explained later.<sup>13</sup> The SDM has gone away, and will be superseded by the proposed “DICOM Content Mapping Resource” (DCMR).<sup>14</sup>

## Pros and Cons of Codes

As one can see from the encoding of the triplet, compared to simple strings, coded entries are more complex to parse and expand the size of the data set. However, there are tremendous advantages to using coded entries, including:

- ▶ the ability to indirect lists of defined terms and enumerated values through separate tables
- ▶ the potential for reuse of common lists of terms by different attributes
- ▶ the potential to “out-source” the definition and maintenance of lists of terms to groups with greater expertise or resources
- ▶ the potential to achieve consistency of terminology between different specialities, industries and geographic regions

A classic example quoted in this respect is the use of anatomical terms. Radiologists need to be able to refer to the body part that is the foot, just as do pathologists, podiatrists and orthopedic surgeons. For that matter, English speaking, French speaking and Japanese speaking radiologists all need to refer to the concept of a “foot” but would encode it as a string quite differently. If each country, speciality and message standard used its own strings, then there would be no possibility for mutual understanding.<sup>15</sup> By reaching consensus on the meaning of a concept and how to encode it, one potentially achieves both better understanding and reuse. There will always be a requirement for mappings between competing or historically prevalent (often regionally or site specific) coding schemes.<sup>16</sup> There will also remain domain or nationally specific requirements, such as for codes for reimbursement.<sup>17</sup> The use of coded entries rather than arbitrary strings facilitates, but does not guarantee, progress in interoperability.

---

13. This change was made in Supplement 36.

14. The DCMR will be PS 3.16 of DICOM, proposed in Supplement 53.

15. At least not without a combinatorial expansion of mappings between each coding system.

16. In the US, the National Library of Medicine (NLM) sponsors the Unified Medical Language System (UMLS) project to build just such a mapping.

17. For example, in the US, Current Procedural Terminology (CPT) codes to specify procedures are required by the federal government for reimbursement.

## **Coding Schemes to Choose From**

There are many possible coding schemes to choose from. Ideally there would be one coding scheme that everyone in the world could agree on, or a scheme that incorporated and mapped everybody else's coding schemes. The latter is one of the goals of the SNOMED coding scheme from the College of American Pathologists (CAP). Early on, DICOM agreed to work with CAP to incorporate SNOMED codes into DICOM where possible, and to contribute new codes as new applications arose.<sup>18</sup> At the present time, many of the codes specified for use in DICOM image objects are from SNOMED. These are identified with a coding scheme designator of "SNM3" (or "SRT" for more recent codes).

SNOMED is not all inclusive however, and furthermore, DICOM users do not have the right to use any code they please without paying royalties or license fees, only a negotiated subset.

The notion of having to pay fees to use codes may seem odd to some people. The situation seems to be analogous to licensing of software or fonts. Communications standards like DICOM or HL7 can be used without paying a fee, though one may have to pay to get an official copy of the documentation. Coding schemes tend to be distributed freely, but one has to pay to actually use them. Interestingly, the business model for the use of codes seems to favor actual users paying the fees, rather than equipment or software vendors. Typically a hospital or healthcare enterprise pays to use the codes it needs, often on multiple vendors' systems. Contrast this model with license fees for the use of a patented compression or encryption scheme used in a device, which a vendor would usually pay.

The DICOM position on having to pay for codes is that since one shouldn't have to pay to use DICOM, the standard can't mandate the use of any codes that require royalty fees. There is a distinction between codes that are:

- ▶ mandatory
- ▶ optional, but required for basic interoperability
- ▶ optional, but not required for basic interoperability

For example, in the digital X-ray and mammography family of image objects, some of the anatomical codes are mandatory. In these cases a particular code from a particular coding scheme must be chosen, and never change. These are "enumerated values." In other cases, a list of defined codes is needed to be interoperable, such as for view position. That is, everyone needs to use the same code for say, an antero-posterior projection. Though one can extend the list to describe new or unanticipated views, one is not allowed to substitute a different code (whether it be from the same or a different coding scheme) for a concept that is already defined. These are "defined terms."<sup>19</sup>

---

*18. This agreement has recently been formalized, allowing users of the DICOM standard to continue to use a subset of the SNOMED codes without license fees or royalties, as well as allowing DICOM to continue to submit new codes. Strictly speaking these codes are not "free" but rather are "royalty pre-paid," the notion being that DICOM contributed work on these codes in lieu of royalties. The lawyers like this concept better.*

Those codes which are mandatory, and those which are optional but required for basic interoperability, all need to be free for use in DICOM. This constrains the standard to choose only those codes for which a royalty pre-paid license can be negotiated, like the SNOMED subset, those which are completely free, like LOINC<sup>20</sup> or ICD,<sup>21</sup> or those that are actually created and maintained by DICOM itself. The DICOM Committee really wants to avoid having to construct and maintain lists of codes itself, both to reduce the workload and to prevent inconsistency with other standards groups. Inevitably however, there will be some concepts that arise strictly within the DICOM domain that have not yet been modeled by other coding groups, or cannot be incorporated in a timely fashion.

Table 2 lists some of the coding schemes that have been identified as containing unique codes for various useful concepts. This table is an amalgam of coding schemes listed in HL7 Version 2.3.1, ASTM E1238-97,<sup>22</sup> a similar list that was formerly contained in Annex D of DICOM PS 3.3, and coding schemes that have been more recently adopted by DICOM. The distinction between diagnostic as opposed to other coding schemes has been elided, sources have been abbreviated, and comments added.

There are a number of things to notice about this table:

- ▶ there is a lot of overlap in territory between different coding schemes
- ▶ some coding schemes are highly nationally specific
- ▶ many coding schemes are not likely to be directly relevant to DICOM and structured reporting
- ▶ historically, “99SDM” is sometimes used in DICOM rather than “SNM3” to identify some codes used in older objects; “SDM”, “SNM” and “SDT” should never be used in DICOM objects; more recent SNOMED codes will be drawn from “SRT”
- ▶ some references to DICOM in this list, such as “DCL” and “DQL”, are not currently defined by DICOM, and should probably never be used; codes defined by DICOM are always identified by “DCM”
- ▶ codes from the CEN SCP-ECG standard as used in DICOM are identified as “SCPECG” rather than “CE”; “CE” is specified in ASTM E1238, but the means of defining code values is not the same as is used in DICOM

Notice also that the addition of future schemes to a table like this is in question. HL7 is apparently planning on switching to using ISO Object Identifiers (OIDs) in future versions, which are like DICOM UIDs, rather than simple short string codes.<sup>23</sup> This is a problem for DICOM because the value representation for Coding Scheme Designa-

---

19. Strictly speaking, the nomenclature of “enumerated values” and “defined terms” applies only to traditional DICOM attribute values. The corresponding nomenclature for the value sets of coded entries are “enumerated context groups” and “defined context groups,” as will be explained later.

20. Logical Observation Identifiers, Names, and Codes

21. International Classification of Diseases.

22. An earlier version of which seems to have been the original source of this list.

tor is only 16 characters long.<sup>24</sup> DICOM has already begun to make use of coding schemes not in the tables specified in HL7. It is unclear whether or not these will be incorporated in future HL7 or ASTM tables.

Out of this long list, probably the most likely to be encountered in DICOM structured reports will be:<sup>25</sup>

- ▶ SNOMED (“SNM3” and “SRT”)
- ▶ ACR Index for Radiologic Diagnosis (“ACR”)
- ▶ ICD9 and ICD10 (“I9” and “I10”)
- ▶ ICD9-CM and ICD10-PCS (“I9C” and “I10P”)
- ▶ DICOM defined codes (“DCM”)
- ▶ UCUM for units of measurements (“UCUM”)
- ▶ CEN SCP-ECG (“SCPECG”)
- ▶ BARI coronary artery segments (“BARI”)

TABLE 2. Coding Schemes listed in HL7, ASTM and/or DICOM

Name	Scheme	Source	Comments
SNOMED/DICOM Microglossary	99SDM	Bidgood	obsolete; historical use for enumerated values in nuclear medicine IOD
Local general code	99zzz	Locally defined	site specifies zzz; not globally unique
ACR Index for Radiological Diagnosis	ACR	ACR	
WHO Adverse Reaction Terms	ART	WHO	adverse drug reactions
ASTM E1238/E1467 Universal	AS4	ASTM	lab procedure codes
AS4 Neurophysiology Codes	AS4E	ASTM E1467	
American Type Culture Collection	ATC	ATCC	microorganisms, tissue cultures
Bypass Angiography Revascularization Investigation	BARI	Coronary Artery Disease 1992,3:1189-1207	codes for coronary artery segments
CPT-4	C4	AMA	procedure codes
CPT-5	C5	AMA	procedure codes
Chemical Abstract Codes	CAS	US Pharmacopoeial Convention	chemicals and generic drugs

23. Using a unique identifier allows non-overlapping identification of local coding schemes, which is an important consideration as hospitals merge and coalesce into larger enterprises.
24. Some sort of macro or entity reference mechanism will probably be used in XML instantiations of HL7 V3, alleviating this problem by allowing shorter strings to substitute for the longer ISO OIDs.
25. Supplement 50, Mammography CAD, proposes to make use of terms defined in the ACR Breast Imaging Reporting and Data System (BI-RADS), but very likely these will be incorporated in SNOMED-RT and be designated as “SRT”, so there will not be a separate “BIRADS” coding scheme designator.

TABLE 2. Coding Schemes listed in HL7, ASTM and/or DICOM

Name	Scheme	Source	Comments
CDT-2	CD2	ADA	dental procedure codes
CDC Analyte Codes	CDCA	CDC	
CDC Methods/Instruments Codes	CDCM	CDC	
CDC Surveillance	CDS	CDC	public health
CEN SCP-ECG diagnostic codes	CE	CEN TC 251 PT 007	not used in DICOM; see SCPECG
CLIP	CLP	Beth Israel	codes for radiology reports
COSTART	CST	FDA	adverse drug reactions
CDC Vaccine Codes	CVX	CDC	
DICOM Class label	DCL		not used in DICOM; in HL7 2.3.1 list
DICOM (Modality) codes	DCM	DICOM	codes defined in DICOM standard
DICOM Query label	DQL		not used in DICOM; in HL7 2.3.1 list
Euclides AFP codes	E	Euclides Foundation	
Euclides kind of quantity codes	E5	Euclides Foundation	
Euclides lab method codes	E6	Euclides Foundation	
Euclides lab equipment codes	E7	Euclides Foundation	
Enzyme codes	ENZC	IUBMB	
First DataBank Drug Codes	FDDC	First DataBank	
First DataBank Diagnostic Codes	FDDC	First DataBank	drug-diagnosis interaction checking
FDA K10	FDK	FDA	device & analyte process codes
HIBCC	HB	HIBCC	business communication
Home Health Care	HHC	Georgetown Univ.	
Health Outcomes Institute codes	HI	Stratis Health	outcome variables
HCPCS	HPC	HCFA	procedure codes
Gabrieli Medical Nomenclature	G	ECRR	
ICD10	I10	WHO	diagnosis codes
ICD10-PCS	I10P	HCFA	procedure codes
ICD9	I9	WHO	diagnosis codes
ICD9-CM	I9C	CPHA	procedures and diagnostic tests
ISBT	IBT	ISBT	blood transfusion (blood groups)
ICHPPC-2	IC2	WONCA	ICD9 modified for primary care
ICD for Oncology	ICD0	WHO	

TABLE 2. Coding Schemes listed in HL7, ASTM and/or DICOM

Name	Scheme	Source	Comments
ICD for Sleep Disorders	ICDS	American Sleep Disorders Association	
ICCS	ICS	CPHA	
IUPAC/IFCC Property Codes	ICU or IUC or IUPP	IUPAC/IFCC	chemical properties (lab) (ASTM says ICU, HL7 says IUC or IUPP)
IUPAC/IFCC Component Codes	IUPC	IUPAC/IFCC	analytes (lab)
Japanese Chemistry	JC8	JACP	lab exam classification
Local general code	L	Locally defined	obsolete; not globally unique
Local billing code	LB	Locally defined	obsolete; not globally unique
Logical Observation Identifier Names and Codes (LOINC)	LN	Regenstrief Institute	laboratory and clinical measures
Medicaid	MCD	HCFA	billing codes
Medicare	MCR	HCFA	provider codes
Medispan Diagnostic Codes	MDDX	MediSpan	drug-diagnosis interaction checking
Medical Economics Drug Codes	MEDC	Medical Economics Data	drug identification
Medical Dictionary for Drug Regulatory Affairs (MEDDRA)	MEDR	Medicines Control Agency, UK	
Medical Economics Diagnostic Codes	MEDX	Medical Economics Data	drug-diagnosis interaction checking
Medispan GPI	MGPI	MediSpan	drug identification
CDC Vaccine Manufacturer Codes	MVX	CDC	
NANDA	NDA	North American Nursing Diagnosis Association	
National Drug Codes	NDC	FDA	drug, dose, manufacturer, packaging
Nursing Interventions Classification	NIC	Univ. Iowa	
National Provider Identifier	NPI	HCFA	
Omaha	OHA	Omaha Visiting Nurse Association	
POS	POS	HCFA	place of service codes (for claims)
Read Classification	RC	NHS UK	drug, procedure, diagnostic codes
CEN SCP-ECG diagnostic codes	SCPECG	CEN TC 251 PT 007; AAMI version 1.3	used in DICOM for waveforms

TABLE 2. Coding Schemes listed in HL7, ASTM and/or DICOM

Name	Scheme	Source	Comments
SNOMED/DICOM Microglossary	SDM	Bidgood	not used in DICOM
SNOMED 2nd ed	SNM	CAP	not used in DICOM
SNOMED V3 1993	SNM3	CAP	most commonly used in DICOM
SNOMED Topology Codes	SNT	CAP	anatomic sites; not used in DICOM
SNOMED-RT	SRT	CAP	used in DICOM for new codes
Uniform Clinical Data Systems (UCDS)	UC	HCFA	
Unified Code of Units of Measure	UCUM	Regenstrief Institute	used in DICOM for all units
Medical Device Nomenclature System (MDNS)	UMD	Universal MDNS	
Unified Medical Language	UML	NLM	
Universal Product Code (UPC)	UPC	Uniform Code Council	
UPIN	UPIN	HCFA	physician identification numbers
WHO record number drug codes	W1	WHO	6 digit drug codes
WHO record number drug codes	W2	WHO	8 digit drug codes
WHO record #, ASTM extension	W4	ASTM	drug codes
WHO ATC	WC	WHO	hierarchical classification of drugs

In addition to “standard” coding schemes, locally defined or private coding schemes are often used. Individual institutions often make use of well-established local schemes for coding procedures and diagnoses. Private schemes are also often appropriate for research projects. According to HL7 and ASTM convention, local schemes are either designated with “L” or “99xxx”, where the value of xxx serves to distinguish one local coding scheme from another. For example, the coding scheme designator “99PMP” is used in this book to indicate codes that are not drawn from a standard scheme. There is no registry of local coding schemes, nor any guarantee that a particular value of “99xxx” is globally unique.

## Codes for Structured Reporting

For general purpose reporting applications, codes are needed for:

- ▶ names for reports for use as document titles
- ▶ standard codes for procedures and protocols that are the subject of reports
- ▶ headings and sub-headings for structuring reports

- ▶ typical conclusions and diagnoses
- ▶ typical findings and measurements
- ▶ units for common measurements

Fortunately, many of these are either already included in SNOMED, LOINC or other coding schemes, have been submitted for inclusion, or will be defined by DICOM itself. SNOMED already contains many codes for general purpose concepts, and LOINC already contains some useful report headings. For example, there is a LOINC code that means “diagnostic imaging report.” Unfortunately, as a consequence of LOINC’s historical basis in coding laboratory tests, the rules for generating “fully specified names” that might be used for Code Meaning are some what arcane. For example, the meaning of “18748-4” is not “diagnostic imaging report” but rather “REPORT:FIND:PT:XXX^PATIENT:NAR:DIAGNOSTIC IMAGING”. If LOINC codes are going to be used in DICOM, then it is likely that the context groups that specify them will also need to include suitable synonyms that may be used.

Standard codes for document titles, headings, sub-headings and corresponding individual content item concept names like “findings” and “finding,” “conclusions” and “conclusion” are expected to be defined in fairly short order, as are codes for common concepts like “normal.”

Units for numeric measurements are a special case. There are two fundamentally different approaches to creating codes for units. One is to define a long list of pre-coordinated codes for every conceivable measurement, and assign some arbitrary (unparseable) code. This is the traditional SNOMED approach,<sup>26</sup> and a unit like “milliliters per second” might be represented by a code such as “Y-U0222”.<sup>27</sup> An alternative approach is to “construct” codes from atomic components according to a set of rules. The latter approach is adopted by the Unified Code for Units of Measure (UCUM), which attempts to unify previous efforts at defining codes for units of measure.<sup>28</sup> UCUM specifies atomic components and lexical rules for combining them to produce more complex units. For example, one might create a unit of flow from units of volume and time, such as milliliters per second. In the case-sensitive form of UCUM encoding, this would be represented as the string “ml/s”. To use UCUM in DICOM, a few additional rules are required, such as to specify whether the case-sensitive or case-insensitive value will be used in the Code Value attribute, and how to construct meaningful names to encode in Code Meaning.<sup>29</sup> The use of a “constructive” scheme like UCUM does not preclude one from specify a short list of predefined codes for a particular purpose.<sup>30</sup> Rather it defines a logical, extensible and understandable scheme

---

26. *Though SNOMED does not yet address units of measurement, it plans to in the future.*

27. *“Y-U0222” is not a real code, so don’t use it. By the way, any codes discussed in relation to SNOMED that start with a “Y” are temporary codes that were proposed in the SNOMED-DICOM Microglossary. The intent was that permanent codes would later be assigned but in some cases this didn’t happen. One should not use any codes starting with “Y” since they are not actually in SNOMED.*

28. *Specifically, it draws on ISO 2955-1983, ANSI X3.50-1986 and HL7’s extensions called “ISO+.” It can be found on-line at “<http://aurora.rg.iupui.edu/UCUM/>”.*

for building such lists in the first place, without the need for an unwieldy master list of every potential permutation and combination in a pre-coordinated form.

## **What does a Code Represent?**

Codes have been described without actually considering what each code signifies. A code could simply be an alternative representation of a particular string, a so-called “string code.” Such a one to one correspondence between a code and a string can do no more than substitute one representation for another. In the old days, when compactness of representation was a factor, this was one of the primary reasons for the use of codes. Obviously since the triplet coding scheme significantly expands the representation in DICOM, there must be more to it than this.

The codes used in DICOM actually represent concepts rather than simple strings. This allows the same code to be used even if there are multiple synonyms which are merely different string representations of the same concept. For example, “craniad” and “cephalad” are two alternative ways of saying “closer to the head.” That is, they both represent exactly the same concept.

The use of coded concepts also achieves independence from annoying inconsistencies in string representations, such as capitalization, hyphenation and abbreviation. For example, “Medio-lateral Oblique”, “mediolateral oblique” and “MLO” are all alternative representations of the same mammographic projection, and mean precisely the same thing.

Conversely, two entirely different coded concepts may have similar or identical string representations. For example, one of the string representations for the Law radiographic projection may be “Law”, and one of the string representations for a rule from the legal system may be “Law”, but these are clearly two entirely different concepts and would have different concept codes. Another example is the string “mole”, which may be a unit of measurement of the amount of a substance, or an animal that burrows underground.

When one starts to consider the context in which a coded concept might be used, things start to get a little more difficult. The term “lateral” is widely used in medical applications, for example to describe the notion of being further away from the midline and towards the side (i.e. the opposite of “medial”). It is also used to describe a radiographic view or projection that is “from the side”. These are related concepts, have the same string representation, but are probably sufficiently different to deserve different concept codes. This is certainly true if one wants the concept of a “lateral view” to stand alone, outside the context implied by a name-value pair like “view=lat-

---

29. *What has been proposed is to allow synonyms consisting of either the same meaning as the value (i.e. an abbreviation), or the full names either using the US or European convention for spelling (i.e. either “meters” or “metres”). For example the meaning of the code “ml/s” could be “ml/s” or “milliliters/second” or “millilitres/second”.*

30. *For example, the set of codes for units of measurements defined for annotation of waveforms in Supplement 30 consists of a such a short list derived using UCUM rules.*

eral". Likely then, the concept would be modeled as "lateral view" but have a commonly used synonym of "lateral".

Coded concepts also present opportunities for international harmonization. Though there is some regional variation in the interpretation of similar concepts, most of the variation is in gratuitously different string representations of the same concepts. Thus the "breast" as an anatomic region means the same to someone in France, even though it may be written "sein". In Italy it might be written "seno", in German, "Brust", in Japan "乳房" or "にゅうぼう", etc. In DICOM, thanks to SNOMED, it is written ("T-04000", "SNM3"), regardless of the locale. Depending upon the application, the code meaning conveyed in the message as part of the triplet code may not be localized, but a receiver with a localized dictionary may look up the code to find a localized string equivalent. In theory, a report that consisted entirely of codes without plain text would be understandable to anyone, regardless of the language they spoke, as long as they were in possession of a localized dictionary.

Whether or not one can send localized strings in the message as the value of code meaning attributes is a little controversial. It is desirable to avoid everyone making up string synonyms, so the standard specifies that the code meaning must be taken from the coding scheme. This does not mean it must be in English, but unfortunately there are as yet few, if any, non-English translations of the coding schemes used most in DICOM. One solution is to make such translations and have them incorporated into schemes like SNOMED. DICOM has negotiated the right for its users to do this. Regardless, it is clearly unacceptable in regions where the primary language is not English to expect everyone to encode English strings. Indeed in some countries it is probably illegal to insist on displaying one language alone, especially when that language is English!

The localization issue has been less of a problem with image objects, since relatively few codes are in use. It is quite straightforward to localize a small dictionary to map encoded attributes into on-screen representations. Structured reporting, on the other hand, has the potential to greatly exacerbate this problem. This is particularly true for naive report rendering applications that would prefer to just "splat" the transmitted code meaning directly onto the paper or screen. More sophisticated applications that do de-reference the code through a lexicon still have the problem of choosing which synonym to use.<sup>31</sup>

A further problem arises when one considers the need for bi-lingual or multi-lingual reports. In some countries, such as Canada, it might be desirable to create an entire report that contains equivalent meanings in say, French and English. Since there can only be one code meaning actually transmitted for a particular instance of a coded entry a different approach is required. This will be discussed further in the chapter on "Encoding & Relationships: Building the Tree," specifically when discussing the use of

---

31. For example, when rendering a name-value pair like "view=lateral", one might prefer to render the value of the code for "lateral" as "lateral" rather than the synonym "lateral view", to avoid the ugliness of "view=lateral view". Making this choice requires greater understanding of the meaning of the codes, and in the simple case, might be better made by the creator of the object.

the “has concept modifier” relationship. The subject of internationalization in general is covered in the chapter on “Internationalization and Localization.”

## Code Meaning Revisited

From the foregoing it is apparent that a coded entry is a representation of a concept, and that there may be multiple ways to render the meaning of that concept as a consequence of:

- ▶ synonyms
- ▶ different languages

There are additional factors that impinge upon the choice of a value to encode in the code meaning field of a coded entry, and the choice of meaning to actually render on the screen or on paper.

For example, there are questions of punctuation, capitalization and word order. Typically, the meaning for a concept will be in a first letter capitalized, hyphenated form like “Antero-medial” rather than an alternative like “anteromedial”. This is not always the case however, nor is there great consistency amongst the lexicons or even within one lexicon in this respect. According to the letter of the law, one must use one of the meanings from the coding scheme indicated by the coding scheme designator; the encoder cannot just “coerce” punctuation and capitalization as it sees fit. Of course, the rendering device can clean this up in any way it likes, so long as the meaning does not become ambiguous. The issue of word order arises for codes that are pre-coordinated, that is for codes that represent both a generic concept and its qualifiers or modifiers, such as “Knee, left”. The encoder might prefer to send this as “left knee”, but unless such a synonym is present in the coding scheme, it really should not make this substitution.

What is the reason for this restriction? The answer would be obvious if there was always a one-to-one correspondence between the code value and the code meaning, in which case the code would merely represent a string and not a concept. Since the codes do represent concepts and not just strings, and there may be synonyms for a concept, would there be any harm in allowing an encoder to create its own synonym as it saw fit? There is concern that there not be a divergence of interpretation of what concept a code actually represents. The creators of a lexicon, presumably experts in their field, include appropriate synonyms. If every user of a code were to use their own “creative” interpretation of a coded concept to produce their own synonyms, they might inadvertently diverge from the intended definition of the concept.

Another concern, particularly for pre-coordinated codes that might have very long descriptions, is the 64 character limit on the length of the code meaning in DICOM.<sup>32</sup> The limit is a consequence of the choice of the Long String (LO) value representation,

---

32. Note that the limit is 64 characters, not 64 bytes. The length in bytes may be considerably longer than 64 bytes if specific character sets with ISO 2022 escape sequences are in use. Implementers beware, do not overflow buffers and dump core when encountering more than 64 bytes in LO values.

and is not something that can be changed, since codes are already used by existing objects in DICOM. One solution is to avoid using heavily pre-coordinated codes, and instead construct an equivalent meaning using the SR tree with the “has concept modifier” relationship. For those cases in which the lexicon does define an extremely cryptic short meaning that will fit in 64 characters, either:

- ▶ the sender can rely upon the receiver to look up the code in its own lexicon and find a more descriptive meaning, or
- ▶ in a structured report, the sender can qualify the meaning with a “has concept modifier” child that contains a value with a longer plain text meaning, and a concept name that says so

A good example is the set of LOINC codes, which mostly defines names of laboratory tests. The LOINC codes are pre-coordinated, and each code represents multiple properties of an individual test. These can be particularly unwieldy, not to mention cryptic, when abbreviated to 64 characters. For example, the meaning corresponding to the LOINC code value 9375-7 is “GLUCOSE^2H POST 100 G GLUCOSE PO:MCNC:PT:SER/PLAS:QN”. This is a pre-coordinated code that specifies a “glucose tolerance test.” This particular code represents a test that is performed by measuring the serum glucose two hours after oral administration of 100 grams of glucose, as opposed to another code that might represent a different glucose tolerance test with different amounts of substrate administered with different timing.

The discussion of the 64 character limit begs the question of where the meanings of an appropriate length can be found. The best answer is that the mapping resource that specifies the use of the selected codes from a particular coding scheme should define these.<sup>33</sup> A problem arises if one wants to use a code from a coding scheme that is not contained in such a mapping resource, and the “native” meaning described in the coding scheme exceeds 64 characters. Some coding schemes don’t actually define a single string meaning. This is the case, for example, for ICD10-PCS procedure codes. In such cases, the most appropriate approach may be to define one’s own “private” mapping resource that defines a suitably short meaning, and document it in the DICOM conformance statement for the product.

## **Not Otherwise Specified**

One will sometimes encounter the rather strange suffix “, NOS” appended to the meaning of a code. This stands for “not otherwise specified.” For example, one might see a code with a meaning “Hand, NOS”. This means that the reference is to the hand, without further specifying whether it is the left hand or the right hand, or some selected part of the hand. Most such concepts have synonyms without the suffix “,

---

<sup>33</sup> In the case of those codes defined for use in the DICOM standard, this will be PS 3.16 of DICOM, the DICOM Content Mapping Resource (DCMR).

NOS”.<sup>34</sup> Rendering applications would probably do well to elide the “, NOS” when displaying or printing the meaning of the code.

## **Context Groups and Mapping Resources**

Earlier, when discussing sources of codes, mention was made of “lists” of codes that were “enumerated” or “defined.” In DICOM, a list of codes from which a code may be chosen is referred to as a “context group.”<sup>35</sup> In the definition of a code sequence attribute<sup>36</sup> there are four possibilities:

- ▶ there are no constraints, that is, no context group is specified; the creator of the attribute can use any code from any coding scheme that is meaningful
- ▶ a list of codes is suggested for use, but alternative codes with the same or similar meaning may be used at the discretion of the creator; this is referred to as a “baseline context group”
- ▶ a list of codes is specified, but the list may be extended with other codes, as long as they do not overlap in meaning with the specified codes; this is referred to as a “defined context group”
- ▶ a specified list of codes shall be used, and no other codes may be used; this is referred to as an “enumerated context group”

For those familiar with traditional DICOM string attributes, the concepts of “defined context groups” and “enumerated context groups” correspond to “defined terms” and “enumerated values” respectively. There is no historical equivalent of the “baseline context group” for string values in the existing standard.

A context group is referred to as baseline, defined or enumerated *where it is used*, not where it is defined. Thus, the same context group may be used in some objects or templates as a defined context group, and in other places as an enumerated context group. This design facilitates the reuse of useful lists of terms in different circumstances.<sup>37</sup>

The context groups are not defined in the coding schemes where the meanings of the codes themselves reside. Rather, context groups are defined in a “Mapping Resource,” which “maps” lists of codes from where they are used, in context, to the schemes in

---

34. The synonym with the appended “NOS” may in some cases be “inactive” or “deprecated.” More recent versions of SNOMED, for example, appear to avoid the use of a “not otherwise specified” suffix.

35. The origin of the term “context group” lies in the fact that one list of codes or another may be more appropriate, depending on the “context” in which the code is used. The term “value set” is also used to describe the set of values that an attribute may take, which in the case of a coded entry, is described by reference to a “context group.” See the chapter on “Templates: Bringing order to chaos ...” for further discussion of value sets.

36. A “code sequence attribute” is one that is encoded as a coded entry. That is, it is an attribute with a VR of SQ whose items are specified by invoking the code sequence macro.

37. Accordingly, one must be very careful when adding terms to existing context groups to make sure that they have never been used as enumerated context groups.

which they are defined. One of the consequences of this “indirection” of context groups through mapping resources is that a single context group may contain codes drawn from *different coding schemes*. For example, a single context group defining a list of codes for a certain class of procedures might contain codes from SNOMED, LOINC and ICD9-CM, if one coding scheme alone was insufficient to describe all the concepts required.

Historically, the first mapping resource used in DICOM was the SNOMED-DICOM Microglossary (SDM).<sup>38</sup> The SDM has been superseded, and its function will be subsumed by the DICOM Content Mapping Resource (DCMR), PS 3.16 of the standard.<sup>39</sup> The original SDM was used to organize anatomical terms, positions, projections and lists of devices, drugs and interventions, for use in the “second generation” of DICOM image objects (specifically, ultrasound, nuclear medicine, X-ray angiography and radiofluoroscopy (XRF)). The SDM underwent considerable evolution beyond that point, but ran into trouble as a consequence of concerns over:

- ▶ where the SDM would actually be located and who would maintain it
- ▶ whether or not the SDM was to be balloted as part of the DICOM standard (a requirement of mandating its use in any particular attribute or object)
- ▶ whether or not the SDM would include codes from coding schemes other than SNOMED
- ▶ whether or not it would include codes that would require DICOM users to pay license or royalty fees

Subsequently, based on an agreement with the SNOMED Authority, the SDM “evolved” into the DCMR. The DCMR:

- ▶ will be formally maintained by DICOM
- ▶ will be part of the official balloted published DICOM standard
- ▶ will use codes from any suitable source, but with a preference for codes available from a royalty pre-paid subset of SNOMED
- ▶ will require no payment of license or royalty fees by users of the DICOM standard

As a consequence of this evolution, the DCMR will be somewhat less comprehensive than the SDM. In particular, it will no longer include the bulk of anatomical codes from SNOMED, only a small subset for selected context groups. The DCMR will be sufficient, however, to meet the needs of the currently defined DICOM objects. Furthermore, codes for new concepts developed by DICOM will be submitted to SNOMED (or other appropriate sources of codes) for inclusion and subsequent use without fee (royalty pre-paid). This process is intended to be the primary mechanism for the addition of new concepts to the mapping resource.

---

38. The SDM was created and maintained by Dean Bidgood with an enormous investment of his time and energy, and sponsored by the College of American Pathologists.

39. The DCMR is proposed in Supplement 53.

## Context Groups and Synonyms

Context groups have been described as a means of specifying an attribute's value set. There is a little more to context groups than might be apparent at first sight. If context groups were no more than lists of terms then they would probably be called "lists of terms". What does the word "context" in the name imply?

The implication is that the interpretation of the meaning of a term may vary depending upon the "context" in which it is used.

For example, there is a code T-56000, for the "esophagus". One might wish to say that the "anatomic region of interest" is the esophagus, perhaps for a diagnostic procedure like an upper gastrointestinal endoscopy. On the other hand, one might be examining the heart during an echo-cardiography procedure, in which case the "transducer position" may be *within* the esophagus.<sup>40</sup> In the case of the endoscopy one might write:

```
Anatomic Region = (T-56000,SNM3,"Esophagus")
Transducer Position = (T-56000,SNM3,"Esophagus")
```

whereas in the case of the trans-esophageal echo-cardiography (TEE), one might write:

```
Anatomic Region = (T-32000,SNM3,"Heart")
Transducer Position = (T-56000,SNM3,"Esophagus")
```

Notice here that the same code is being used but in different "contexts." If one looks at the code alone, irrespective of context, such as while searching the document, one does not get the whole picture.

The use of the same code in different contexts also gives rise to a need for synonyms. For example, it is common parlance to say that a transducer is "endo-esophageal" rather than "in the esophagus". While one might argue that different codes should be assigned for the more specific concepts "in the esophagus" and "about the esophagus," this is not always the case.<sup>41</sup> The code "T-56000" was chosen for this example because "endo-esophageal" is defined in the coding scheme as a synonym.<sup>42</sup> Accordingly, instead of:

```
Transducer Position = (T-56000,SNM3,"Esophagus")
```

one might write:

```
Transducer Position = (T-56000,SNM3,"Endo-esophageal")
```

40. Those not familiar with echo-cardiography may well ask why would one want to examine the heart from within the esophagus rather than outside the body. The esophagus is a lot closer to the heart, and the lungs are not in the way to attenuate the sound. In other words, for the price of a little discomfort, one gets much better images.

41. And would lead to a combinatorial explosion of such pre-coordinated codes.

42. Indeed "endo-esophageal" rather than "esophageal" is listed in the english nomenclature field for this code in the context group for ultrasound transducer positions. There is also a more specific anatomic code that refers not to the esophagus as a whole but the "esophageal lumen", T-56450, which might be more appropriate and specific in this context.

An application that is rendering the coded entries into something readable in a report might prefer to use the more descriptive synonym. This gives rise to the question as to whether or not an application should use the supplied code meaning, or look up the code in its lexicon. If the latter, how should it choose between synonyms to render? The “correct” answer would be to determine the synonym from context. That is a “non-trivial” task.<sup>43</sup> Avoiding dependence on the code meaning field is important for other reasons, such as localization, as will be apparent later. Using code meanings with specific synonyms is tantamount to saying that the real code is defined by the combination of code value, coding scheme designator *and code meaning*, which is not the intent at all. Ideally, only true synonyms rather than “subtle refinements” would be allowed in the coding schemes.

In order to preserve the “original semantic context” in which a term was selected, additional attributes can be transmitted along with the coded entry. This is the “enhanced encoding mode” which will be discussed in the next section.

However, even in the absence of additional coded entry attributes, there are implicit sources of context including:

- ▶ the attribute name for which the code is a value
- ▶ the module or IOD in which the attribute is used
- ▶ the concept name for which the coded value type is a value
- ▶ the template that was used to build the content
- ▶ modifiers that may be applied

In the simple example given before:

- ▶ if the term for esophagus is used in the context of describing transducer position, the context additionally implies that the transducer is inside the esophagus (“endo-esophageal”)
- ▶ if the term for esophagus is used in the context of describing the anatomic region of interest, the context additionally implies that the region being examined is the esophagus (rather than the heart)

In the case of “traditional” DICOM objects, in which named attributes have specific concepts associated with them (like “Transducer Position Sequence”) then determining the context is relatively easy. The IODs in which such attributes are used often explicitly specify baseline, defined or enumerated context groups. In structured reporting, where the attributes like “Concept Name” are generic, and are themselves coded entries, determining implicit context in this manner involves interpreting the semantics of the concept name. Templates, which will be described in a later chapter, are the analog of IODs for structured reporting. They also may specify baseline, defined or enumerated context groups for the values of concepts. As such, templates are a potential source of implicit context.

---

43. “Non-trivial” is used here in the sense of “almost intractable.”

The use of modifiers is particularly important to consider with respect to context and synonyms. The coded entry macro itself does not describe a mechanism to modify the meaning of a code. However, in both traditional DICOM objects and the SR content tree, there are mechanisms to attach a modifier to a concept. In the ultrasound example, depending on where the concept of “transducer position” is being encoded, it may be possible to specify a modifier that indicates that the coded entry for the position is “intra-cavitary.”

What does this mean in practice for implementers who receive codes and have to struggle to render meaning?

- ▶ naive implementations that ignore context entirely and render the transmitted code meaning literally will work just fine, presuming the sender chose the meaning wisely; they will not work if the code meaning is not appropriate to the locale (e.g. in the wrong language or character set), or if objects are received from systems that don’t transmit code meaning<sup>44</sup>
- ▶ implementations that always consult the lexicon for meaning, independent of context, have no problem unless synonyms exist, in which case they may wish to compare the transmitted code meaning with the available synonyms to make a choice
- ▶ more robust implementations that always indirect their lookup of the code through an additional lexicon of context groups (the mapping resource) will have a more context specific range of synonyms to choose from; determining which context group to consult depends on knowledge of explicit or implicit context, as described earlier<sup>45</sup>
- ▶ the most advanced implementations, those which actually try to construct a grammatically correct natural language statement from the coded entries or attempt to “understand” context-sensitive subtleties, may need to use more complex models of context

In all cases, the presence of modifiers will likely reduce any residual ambiguity.

One way to restate the foregoing more concisely is that the meaning of a code is determined by more than the just a pair (code value, coding scheme designator). Rather it is determined by a tuple (context, code value, coding scheme designator).<sup>46</sup>

---

44. The HL7 equivalent of a coded entry, the CE data type, does not require that code meaning (or “display name” in HL7 parlance) be sent.

45. Specifically, the context group may be determined from attributes of the enhanced encoding mode, or from a priori knowledge of the value set of the concept name and its context.

46. Presented here is a “bottom-up” perspective of coded entries in DICOM, that is, from the viewpoint of an application parsing coded entries and having to do something sensible with them. An alternative, “top-down,” perspective is provided in a paper by Dean Bidgood on the SNOMED DICOM Microglossary, in which he states:

“Context groups are lists of terms or phrases that are instances of a specified concept-type. Individual terms may appear in any number of context groups. This provides a mechanism for representing multiple inheritance in SNOMED without violating the SNOMED Version 3.x ‘one concept, one termcode’ encoding convention. Inclusion of a term in multiple context groups allows multiple senses of context-dependent meaning to be defined with essential characteristics, description, examples, and notes for each instance.” *Meth Inf Med* 1998;37:404-414.

To digress slightly, there may be other reasons why one might want to know from what list a code was chosen. For example, the choice of code to use might be affected by the range of choices available. If a different, larger or more granular set of choices were available, different choices might have been made. For example, when encoding the post-operative sex of a male to female trans-sexual patient, if required to choose the sex from the list “male” or “female”, one might choose “male”. Given a list of “male”, “female” or “other”, one might choose “other”. Given a list of “male”, “female”, “hermaphrodite”, “trans-sexual” or “other”, one would likely choose “trans-sexual”. Conversely, the choice of “male” from the last list gives one much greater confidence that the patient is not a hermaphrodite or trans-sexual, compared to the choice of “male” from the first, shorter list.<sup>47</sup>

The enhanced encoding mode described in the next section, allows explicit identification of the context in which a code was chosen. This mode also allows more precise specification of the version of that context, allowing the list of choices available to be determined in retrospect.

## **Beyond Triplets... Enhanced Encoding Mode**

The three mandatory components of a coded entry have been discussed: code value, coding scheme designator and code meaning. The coding scheme version, which is conditionally required, is in many cases made redundant by the version implicit in the coding scheme designator.

The standard defines some additional attributes of a coded entry that may optionally be encoded. These are useful in three scenarios:

- ▶ to identify the list from which the code was chosen (the “pick list” or context group)
- ▶ to convey whether or not a private or extended mapping resource was used
- ▶ to convey whether or not a private or extended coding scheme was used

The additional attributes of the code sequence macro used in the enhanced encoding mode and their requirements are listed in Table 3.<sup>48</sup>

For example, to specify in a coded entry that a term was chosen from Context Group ID 4013 of the DCMR, one would add the Context Identifier and Mapping Resource attributes to the coded entry, as follows:

```
(0008,0100) Code Value "T-04000"  
(0008,0102) Coding Scheme Designator "SNM3"  
(0008,0104) Code Meaning "Breast"  
(0008,0105) Mapping Resource "DCMR"
```

---

47. Ok, so this is a pretty contrived and tenuous example. Dean Bidgood provides much better examples in his paper on the subject, such as when retrospectively reviewing a choice of antibiotics from a list available when a decision was recorded.

48. The descriptions and conditions are not repeated here for brevity. They can be found in Section 8 of PS 3.3 of the standard.

**TABLE 3.** Summary of Enhanced Code Sequence Macro

Attribute	Tag	Type
Context Identifier	(0008,010F)	3
Mapping Resource	(0008,0105)	1C
Context Group Version	(0008,0106)	1C
Code Set Extension Flag	(0008,010B)	3
Context Group Local Version	(0008,0107)	1C
Private Coding Scheme Creator UID	(0008,010C)	3
Code Set Extension Creator UID	(0008,010D)	1C

(0008,0106) Context Group Version "19980901"  
 (0008,010F) Context Identifier "4013"

Note that one always has to specify a Mapping Resource when using the Context Identifier, otherwise one couldn't tell from which set of context groups the identifier was chosen. One is also required to specify the Context Group Version, since, for baseline and defined groups at least, later versions may contain more terms, and without the version being specified, one wouldn't know exactly from which list the code had been chosen.<sup>49</sup>

## Private and Extended Coding Schemes and Mapping Resources

The other attributes are used to indicate the use of private coding schemes, and private extensions of either coding schemes or context groups.

If the Coding Scheme Designator is a private coding scheme (such as a site specific coding scheme of procedure codes) then the optional Private Coding Scheme Creator UID may be used. For example.

(0008,0100) Code Value "209309"  
 (0008,0102) Coding Scheme Designator "99PMP"  
 (0008,0104) Code Meaning "Screening Mammogram"  
 (0008,010C) Private Coding Scheme Creator UID "1.2.3.4..."

If a private extension to a standard context group is used, then in addition to specifying a Context Identifier, the Code Set Extension Flag is inserted and set to a value of "Y", and Context Group Local Version and Code Set Creator UID are included:

(0008,0100) Code Value "T-A0100"  
 (0008,0102) Coding Scheme Designator "SNM3"  
 (0008,0104) Code Meaning "Brain"  
 (0008,0105) Mapping Resource "DCMR"

<sup>49</sup> In the past, none of the context groups used in DICOM had explicitly specified versions. This is expected to be rectified by Supplement 53. In the example, the date used for the version is the date of the final text of Supplement 32, in which Context Group 4013 was defined.

```
(0008,0106) Context Group Version "19980901"  
(0008,0107) Context Group Local Version "20000522"  
(0008,010B) Code Set Extension Flag "Y"  
(0008,010D) Code Set Creator UID "1.2.3.4..."  
(0008,010F) Context Identifier "4009"
```

In this example, the defined context group 4009 (specified by the digital X-ray image object for a list of anatomic regions) has been privately extended with the code for "Brain". That is, (T-A0100,SNM3,"Brain") is not in context group 4009. Notice that the additional code used is from a standard, rather than private, coding scheme. This does not have to be the case.

If the Code Set Extension Flag is set, but the Context Identifier is absent, this indicates that the standard coding scheme is being extended.<sup>50</sup> One may specify a standard Coding Scheme Designator, but put in a private value for Code Value that doesn't appear in the designated coding scheme. The meaning of the private code is defined by Code Meaning. This is the one case in which the encoded code value does not actually appear in the designated coding scheme. The code meaning then becomes vitally important. Allowing this mode (as a means of encoding proposed or interim extensions to coding schemes) was probably a mistake, given this inconsistency.<sup>51</sup>

For example, if one wanted to add a new anatomic location for aliens with three feet, then one might encode the following:

```
(0008,0100) Code Value "MYCODE1234"  
(0008,0102) Coding Scheme Designator "SNM3"  
(0008,0104) Code Meaning "Middle Foot (Aliens Only)"  
(0008,0107) Context Group Local Version "20000522"  
(0008,010B) Code Set Extension Flag "Y"  
(0008,010D) Code Set Creator UID "1.2.3.4..."
```

Notice how even though the specified code meaning is bizarre and the corresponding code value does not appear in SNOMED, one still uses SNM3 as the coding scheme designator since the intent is to "extend" SNOMED. This inconsistent usage of code meaning is likely to upset parsers that look for the code value in the lexicon specified by the coding scheme designator, and don't find it. As a consequence it may be important to check for the presence of the Code Set Extension Flag.

Note that even though no Context Identifier is specified (it can't be, since the code hasn't been chosen from a context group, and the intent is to extend a standard coding scheme), one still has to specify Context Group Local version, since it is conditionally required on the presence of a Code Set Extension Flag with a value of "Y". This application of this requirement in this situation is probably also a mistake.

---

50. The Code Set Extension Flag is therefore "overloaded" with two meanings: it signals extension of either the mapping resource or the coding scheme.

51. This is not to say that providing a means of extending coding schemes is a mistake, but rather that the method chosen may be flawed.

Though permitted, there don't seem to be many obvious reasons to send a Code Set Extension Flag with a value of "N".

The enhanced mode of encoding coded entries has not been used much yet. There may well be other issues like this that will crop up and hopefully be resolved through correction proposals.

In practice, private coding schemes are very often used *without* the enhanced encoding mode. This is particularly true when an organization uses a locally defined coding scheme for procedures and protocols. Typically, a Coding Scheme Designator of "L" or "99xxx" will be used without further elaboration. When such codes are used, it is very likely that a receiving application will not possess a copy of the private lexicon, and will have to rely on the transmitted value of Code Meaning.

Another issue that arises in the context of private and extended coding schemes and mapping resources, is the relationship with baseline, defined and enumerated context groups. Clearly, since an enumerated context group is fixed and defined in the standard, there is no question of extending it. A defined context group may be extended by the addition of either standard or private codes. This may be done on an ad hoc basis, or in a more formal manner by defining and describing an extended context group (in the application's conformance statement) and possibly using the enhanced encoding mode to reference it. A baseline context group specified in the standard is no more than a suggestion - any code, private or standard, may be used in place of the suggested codes.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ a code (or coded entry) is used to represent a concept, not just a string
- ▶ coded entries consist of the mandatory triplet of code value, coding scheme designator and code meaning
- ▶ the combination of code value and coding scheme designator uniquely identify the concept being represented
- ▶ code meaning is purely for the convenience of the user, and should never be tested in a condition or used as a key
- ▶ code meaning may contain different strings for the same code, depending on the use of synonyms and local language
- ▶ substitution of arbitrary values for code meaning is forbidden (i.e. one can't "make up" one's own code meaning); only those defined for the code in the designated coding scheme may be used
- ▶ code meaning is limited to 64 characters in DICOM
- ▶ coding scheme version is only required if the coding scheme designator does not imply a version, which it very often does
- ▶ codes are grouped into lists called "context groups" defined in a "mapping resource"

- ▶ context groups are more than just lists of terms, they convey the “context” in which a code was used, which may affect the interpretation of its meaning, and the choice of synonym to render
- ▶ a context group may be baseline (suggested but replaceable), defined (suggested and extensible but not replaceable) or enumerated (required and not replaceable or extensible); the choice is designated when the context group is used (invoked in a module or template), not in the mapping resource where the context group is defined
- ▶ the mapping resource used by DICOM will be the DICOM Content Mapping Resource (DCMR), PS 3.16
- ▶ many of the codes that will be in the DCMR are a royalty pre-paid subset of SNOMED, though other coding schemes are also used; in some context groups, codes are selected from multiple coding schemes
- ▶ enhanced attributes may be used in coded entries to specify private or extended coding schemes, to specify private or extended mapping resources, as well as to identify the context group from which the code was chosen

# ***Content Items: Concept Names and Values***

---

In the chapter describing the basic concepts of DICOM Structured Reports, the notion that SR documents consist of a tree of content items was introduced. Here the individual content items will be described.

Each content item is a name-value pair. The name component of this pair is referred to as the “Concept Name.” It is always defined by a code, rather than free text, to facilitate indexing and searching. These concept names are also used to define headings of containers as well as to indicate the purpose of references to images and waveforms.

Each content item may also have a value. Values may be of various types, including plain text, coded values, numeric values with units, person names, dates and times, references to DICOM images, waveforms and other composite objects, as well as spatial and temporal coordinates.

## **Document Title**

The first concept name that a top down parser encounters while reading an SR document tree is the “document title.” The document title is specified in the standard to always be encoded in the root node as a container content item.<sup>1</sup> It will be described later how containers have no value per se, only children, and so the concept name is all that is present in the root content item.

The document title is really just a “heading” like any other heading, and is only important because the notion of a “title” has real-world significance.<sup>2</sup> Since the root node is

---

1. Or more precisely, a content item with a Value Type of CONTAINER.

2. The concept of titles is a relatively recent notion. It was not until 1472 that the concept of a title page for a document was introduced for a Papal Bull. Prior to that time, manuscripts in libraries were indexed by their first line.

encoded in the top level DICOM data set, it is fairly accessible to conventional DICOM query and retrieve services.<sup>3</sup> A typical example of a document title for a report on a diagnostic imaging procedure might be:

```
<CONTAINER:(209074,99PMP,"Diagnostic Imaging Report")>
```

There are several things to notice about this example:

- ▶ the document title is a CONTAINER
- ▶ the concept name conveys the heading that is the document title
- ▶ the concept name is always a coded entry, never free text

Also notice that, in this example, the title is fairly non-specific. A more specific title can be used, for example to indicate that the report is about a pair of plain X-rays of the chest:

```
<CONTAINER:(209077,99PMP,"Chest X-ray,2V,PA,Lat Report")>
```

However, if one were to use this approach, then a very large repertoire of codes for reports would be needed that "shadowed" all possible procedures.<sup>4</sup> Furthermore, it would be very difficult to perform queries based on the document title. Perhaps a reasonable compromise is to use a code for "Chest X-ray Report" or "Projection Radiography Report" (as opposed to "MRI Report", etc.).

To establish consistency, it is important that a standard set of useful document titles be defined and widely used. This is the subject of ongoing work by DICOM and other groups. It is obviously desirable, for example, that all vendors and users use the same code for a "Diagnostic Imaging Report" whenever possible.

To avoid proliferation of highly specific document titles, the use of post-coordinated rather than pre-coordinated codes may be helpful. The use of the "has concept modifier" relationship to achieve this will be discussed in more detail later. For now some examples will suffice. Instead of:

```
<CONTAINER:(209077,99PMP,"Chest X-ray,2V,PA,Lat Report")>
```

it may be preferable to say:

```
<CONTAINER:(209076,99PMP,"Chest X-ray Report")>  
  <has concept mod CODE:(209100,99PMP,"Views")  
    =(209150,99PMP,"PA and Lateral")>
```

or better still, use various modifiers that qualify an even more general term:

```
<CONTAINER:(209074,99PMP,"Diagnostic Imaging Report")>  
  <has concept mod CODE:(209102,99PMP,"Modality")=  
    (209140,99PMP,"Projection")>  
  <has concept mod CODE:(209103,99PMP,"Region")=
```

---

3. Deeply nested concept names are more difficult to search on, since the DICOM Q/R mechanism has only limited ability to match nested sequence items.

4. That is, one would need a set of codes for every procedure, and another set of codes for a report about every procedure.

```

(T-D3000,SNM3,"Chest")>
<has concept mod CODE:(209101,99PMP,"View")=
(R-10214,SNM3,"PA")>
<has concept mod CODE:(209101,99PMP,"View")=
(R-102CD,SNM3,"Lateral")>

```

A more straightforward approach would be to specify a generic title qualified only by one or more procedure codes:

```

<CONTAINER:(209074,99PMP,"Diagnostic Imaging Report")>
  <has concept mod CODE:
    (209020,99PMP,"Reported procedure")=
    (209300,99PMP,"Chest X-ray,2V,PA,Lat")>

```

A generic search for a particular category of report is considerably easier if all reports use the same code, rather than a plethora of pre-coordinated codes of varying specificity. Similarly it is possible to search for all concept modifiers with a particular code regardless of what is being modified.

## Headings

The document title is a special case of a "heading." A heading is essentially the concept name of a container that has a concept name. One may have containers that are not headings, in which case they don't have concept names. This is a somewhat circular definition, but the principle of a "heading" is fairly fundamental to traditional documents. Organizing ideas into categories,<sup>5</sup> and further nested sub-categories, is second nature to humans and very convenient for machines. This is despite the fact that most people who are not computer scientists don't immediately visualize a pattern of nested headings as a tree (Figure 4).

A traditional report of any kind, be it about a radiological procedure or the annual performance of a company, follows a pattern of structured headings. Accordingly, the design of DICOM SR is heavily weighted towards encoding such a structure easily. Many SR documents follow a pattern that consists of a document title (heading) as the root, with successively nested containers (headings) until one reaches "actual content." For the Basic and Enhanced SR SOP classes, this structure is mandated. In many cases the "actual content" below the last level of heading will be leaf nodes without further children. There is some flexibility in this respect though, depending on the SOP class and whether or not the content is plain text or coded entries.

Consider the following example of a simple plain text report with headings and sub-headings:

```

<CONTAINER:(,, "Chest X-ray Report")>
  <CONTAINER:(,, "Reason for exam")>
    <TEXT:(,, "Reason")="Shortness of breath,
      history of congestive cardiac failure">

```

---

5. In early drafts of Supplement 23, headings actually were referred to as categories, and were the primary reason for the notion of containment.

heading  
 > sub-heading  
 >> sub-sub-heading  
 >> sub-sub-heading  
 > sub-heading

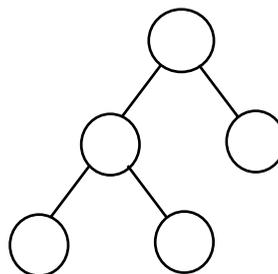


FIGURE 4. Humans see headings, computer scientists see trees ...

```

<CONTAINER:(,, "Description of procedure")>
  <TEXT:(,, "Description")="PA, lateral views
    were obtained">
  <TEXT:(,, "Description")="A left lateral
    decubitus was added">
<CONTAINER:(,, "Findings")>
  <CONTAINER:(,, "Initial Findings")>
    <TEXT:(,, "Finding")="Blunting of the left
      costo-phrenic angle, cardiomegaly
      and interstitial lines">
  <CONTAINER:(,, "Subsequent Findings")>
    <TEXT:(,, "Finding")="Pleural fluid on
      the left">
<CONTAINER:(,, "Conclusions")>
  <TEXT:(,, "Conclusion")="Pulmonary oedema and
    pleural effusion">
  
```

In this simple example, the use of coded entries is confined to the headings. All the actual content is plain text, much as might be dictated by a radiologist. By calling out the headings as coded concept names of containers, the structure of the report is made apparent.<sup>6</sup> The degenerate form of a text only report is much less useful:

```

<CONTAINER:(,, "Chest X-ray Report")>
  <TEXT:(,, "Description")=
    "Reason for exam:
      Shortness of breath, history of
      congestive cardiac failure
    Description of procedure:
      PA, lateral views were obtained
      A left lateral decubitus was added
    Findings:
      Initial Findings:
        Blunting of the left costo-phrenic
        angle, cardiomegaly and interstitial
  
```

6. In this situation, the order of the headings is also important. In structured reports, nested content is always implicitly ordered, as will be discussed later.

```
Lines
Subsequent Findings:
  Pleural fluid on the left
Conclusions:
  Pulmonary oedema and pleural effusion">
```

There is no rule in the standard that says one cannot use this degenerate form of a document title container and one massive block of text as its only child. In the case of legacy reports imported from an existing system, this might well be the best way to make them available in the DICOM environment. However, if the structure of old reports is regular and consistent, as most are, with only a modest effort a few top level headings can be "parsed out" and encoded as container concept names. In particular, separating out the procedure description and conclusion fields as coded entries makes subsequent searches easier.

## Encoding Headings

Now for the hard part, at least for those readers who are not already DICOM aficionados.<sup>7</sup> To actually implement SR, the short hand notation used so far will not suffice. One has to expand this short hand into traditional DICOM data elements with values. A simple case is the document title:

```
<CONTAINER:(209076,99PMP,"Chest X-ray Report")>
```

Since it is a container it has no value, only a concept name. Assuming for now that it also has no children (i.e. is an empty document), it would be encoded as:

```
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209076"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Chest X-ray Report"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
```

At least that is the encoding so far as the document content is concerned; there is also a bunch of stuff up front to identify the patient, study, series and so on, as well as some SR specific management information. The management information is described in the chapter on "Document Management: SR in the Real World." The last example is a minimal, empty, but titled, structured report in DICOM.

In order to add some children, and give the report a little substance, one adds the Content Sequence as follows:

```
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
```

---

7. Or "DICOM weenies" as my wife refers to them (us).

```
(0008,0100) Code Value "209076"  
(0008,0102) Coding Scheme Designator "99PMP"  
(0008,0104) Code Meaning "Chest X-ray Report"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0040,a050) Continuity of Content "SEPARATE"  
(0040,a730) Content Sequence  
    ... child content items go here ...  
(ffff,e0dd) Sequence Delimitation Item
```

Further details of how to nest content are described in the chapter on “Encoding & Relationships: Building the Tree.”

## Value Types

The container just described is one of the most commonly used “value types.” However, containers don’t actually have any value per se, only children.<sup>8</sup> The other value types used in DICOM SR do indeed have values. The following value types are currently defined for use in SR SOP classes:<sup>9</sup>

- ▶ containers (“CONTAINER”)
- ▶ plain unformatted text (“TEXT”)
- ▶ coded entries (“CODE”)
- ▶ numeric measurements with units (“NUM”)
- ▶ person names (“PNAME”)
- ▶ dates, times or dates and times (“DATE”, “TIME”, “DATETIME”)
- ▶ DICOM Unique Identifiers, such as for Study UID (“UIDREF”)
- ▶ references to image objects (“IMAGE”)
- ▶ references to waveforms (“WAVEFORM”)
- ▶ references to other composite objects, such as reports (“COMPOSITE”)
- ▶ spatial coordinates (“SCoord”)
- ▶ temporal coordinates (“TCoord”)

The value type is always explicitly specified. It may be constrained to a limited set of types by the SOP class being used. Templates may further constrain the range of possible value types.<sup>10</sup> Usually, but not always, a template will specify a single value type

---

8. *In a sense, the children of a container are its “values.”*

9. *In future, if it were necessary to add a new value type, say for an array of two dimensional numbers, then the SR framework could be extended, but such a value type could only be used in new SOP classes.*

*The capitalized string in parentheses is the actual value encoded in the attribute Value Type (0040,A040).*

10. *In this sense, template definitions are similar to the module tables of traditional DICOM composite IODs. DICOM SR differs from other IODs in that modules provide only a general framework, and templates are used to describe domain specific restrictions.*

for a particular concept name. Sometimes more than one value type will be allowed for the same concept name. For example, when a physician needs to be identified for some purpose, either a person name or coded value type might be specified.<sup>11</sup>

Content may be multi-lingual, localized and internationalized by using the DICOM specific character set mechanism. Concepts names and values may be indicated by codes that are not specific to any particular country or region. Alternative meanings in different languages for the same concept name may also be explicitly conveyed.

References to external objects such as images and waveforms are constrained to be DICOM objects only. That is, one can reference an instance of a DICOM CT Image Storage SOP class, but not an arbitrary GIF or TIFF file. Similarly, one can reference a DICOM Basic Voice Audio Waveform Storage SOP class instance, but not an AIFF or WAV file. This decision was made in order to constrain SR to a level of interoperability that may be specified within the scope of DICOM by the conformance statement. An SR document can be decoded and rendered without requiring a plethora of unconstrained proprietary plug-ins.

One cannot reference or include pre-formatted text, such as HTML, Word, Postscript or PDF. Only unformatted text may be included, though embedded new lines are permitted. This restriction further separates presentation from semantics.

## Concept Names

Before delving into the details of each value type, a review of the role of concept names is pertinent. Concept names were shown in an earlier section in the role of headings or categories of container value types. For all other value types, the concept name represents the “name” of a “name-value pair.” With only a few exceptions, all the content items in an SR document consist of such name-value pairs.

For instance, one can never just insert an arbitrary chunk of text. Every occurrence of a text value type must have a name as well as a value. So, for example, one cannot say:

```
<TEXT: “Pulmonary oedema”>
```

Rather, one must always specify a concept name as well:

```
<TEXT: (, , “Conclusion”) = “Pulmonary oedema”>
```

This requirement leads to a fairly common pattern of expression in SR documents in which a container with a “plural” concept name is used to group items with a corresponding “singular” concept name. For example:

```
<CONTAINER: (, , “Conclusions”)>  
  <TEXT: (, , “Conclusion”) = “Pulmonary oedema”>  
  <TEXT: (, , “Conclusion”) = “Pleural effusion”>
```

---

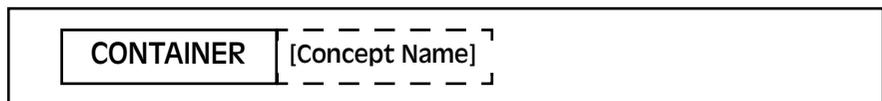
11. The notion of using a coded entry to identify a person might seem strange. Often individuals are labelled with an identifier assigned by some authority, such as a national provider identifier. It is important to convey not only the identifier (as the code value) but also the assigner of the identifier (as the coding scheme designator). The actual name is conveyed as the code meaning. This mechanism is similar to the HL7 2.x XCN data type.

This pattern may seem a little redundant, but it allows each content item to stand alone as a meaningful statement (“conclusion=pulmonary edema”), without having to implicitly “inherit” a concept name from an enclosing container.<sup>12</sup> In many cases, containers become redundant from the perspective of conveying meaning, except to suggest a convenient layout or hierarchy.<sup>13</sup>

## Value Type CONTAINER

The value type of CONTAINER contains no value per se. It may have children; indeed if it doesn’t have any children it is not much of a container.<sup>14</sup> It may have a concept name, but it is not required to. When there is no concept name it is merely a grouping construct.

Graphically, a container value type looks something like this:<sup>15</sup>



As has been described, when a Concept Name is used, in a CONTAINER or any other value type, it is a coded entry, never just plain text. Thus all “headings” in SR documents, which are encoded as containers, are coded entries.

## Continuity of Content

All content items with a value type of container are required to have a Continuity of Content (0040,A050) attribute, which takes a value of either “SEPARATE” or “CONTINUOUS”. This attribute is intended to be a hint to the receiving application that the contents of the container are intended to be rendered either as a list of separate items or as a continuous text flow. In either case, the children of a container are always ordered. There is no way in which to specify an “unordered list” (i.e. that the order of items is not important). The Continuity of Content attribute is only used for containers. It is never present for other value types.

12. The rules for specifying such a hypothetical pattern of inheritance might be difficult to define. Earlier drafts of Supplement 23 did explicitly mention the notion of “observation category context” as a form of observation context that would be inherited. This proposal has been superseded by the notion that the nesting of categories or headings is conveyed by “containment.” That is, a sub-section with its heading is always “contained” in an outer section with its corresponding heading.

13. Applications that render a report as formatted text may feel the need to elide any such redundancy between container headings and enclosed concept names, if it improves readability without introducing ambiguity.

14. Though an empty container might conceivably server as a placeholder to be filled in later, or to convey that a section of a document is empty.

15. The convention that is used here and subsequently is that a solid outline box contains mandatory stuff, and a dashed line outline box with text in “[...]” contains optional stuff. The first box is the value type, the second box is the concept name, and the subsequent boxes are values.

The existence of this attribute is potentially problematic. It is the only attribute in the content tree that is related to presentation rather than semantics. Its inclusion in SR was extremely controversial and the subject of very heated discussion. Be extremely wary of depending on the value of this attribute. It remains to be seen how it will be used in practice.

## Value Type TEXT

A value type of TEXT contains a single text value. The value is encoded as a DICOM Unlimited Text (UT) value representation, which means that it may be as long as is necessary.<sup>16</sup> It is encoded in the attribute Text Value (0040,A160). Despite its “unlimited” length, there are a number of important characteristics of the text that is contained in the value:

- ▶ it contains only a single value, hence the backslash ('\') character is *not* interpreted as a delimiter between multiple values, and may be used in the text
- ▶ it may contain any character in the “default character repertoire” or as defined by the Specific Character Set (0008,0005)<sup>17</sup>
- ▶ it may be padded with trailing spaces, which may be ignored
- ▶ leading spaces are considered to be significant
- ▶ the text is “unformatted,” that is the manner of its presentation for display or printing is not defined
- ▶ the text may contain multiple lines or paragraphs, separated by either LF, CR, CR LF or LF CR;<sup>18</sup> otherwise no format control characters (such as horizontal or vertical tab and form feed) are allowed to be present;<sup>19</sup> whether such a separator is considered to specify a line break or a paragraph break is not specified

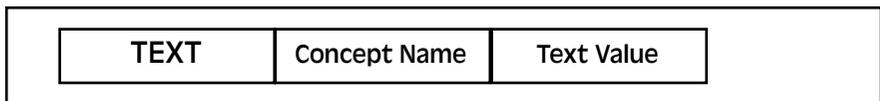
When a content item with a value type of text is used:

- 
16. At least up to a limit of  $2^{n-1}-2$  bytes, which is a consequence of the value length field being only 32 bits long, and the value 0xFFFFFFFF (which is used to mean “undefined length” for some other value representations) being disallowed.
  17. The Specific Character Set (0008,0005) is an attribute of the SOP Common Module that is part of every composite DICOM object (including structured reports and images). It may be used to specify a character repertoire beyond the default (which is US-ASCII). This allows for the use of other single byte character sets, such as the western european ISO IR 100 (ISO 8859-1) set, multi-byte character sets such as Kanji ISO IR 87, and the use of code set extension techniques with escape sequences as defined by ISO 2022. See the chapter on “Internationalization and Localization” for further details.
  18. The characters LF ('\n'), CR ('\r'), and the combinations CR LF and LF CR are considered synonymous. This allows the different new line conventions of different software platforms such as Unix, Mac and Windows to be used interchangeably. Applications must not depend on a particular choice being used; that is, they must behave as web browsers do, and support any of the conventions.
  19. Strictly speaking, the VR allows the use of the FF (form feed) character, but this is forbidden by the definition of the Text Value (0040,A160) attribute itself. The ESC (escape) character is also allowed, but only for use when ISO 2022 code set extension mechanisms are used to switch character sets.

- ▶ it *must* have a concept name, and
- ▶ it *must* have a value with a length of one or more characters.

These restrictions rule out the possibility of inserting plain text that does not have some coded description of its purpose or use; one must say “concept name” = “plain text” and not just “plain text.” Furthermore, one cannot convey the idea of a blank space or placeholder or unknown value by a zero length text value; this is explicitly forbidden.<sup>20</sup>

Graphically, a TEXT value type looks something like this:



As a short hand for describing a text value type, one can write:

```
<TEXT:(,,"Finding")="Large, irregular mass">
```

Or, specifying the concept name more precisely:

```
<TEXT:(209001,99PMP,"Finding")="Large, irregular mass">
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209001"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Finding"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "Large, irregular mass"
```

This looks exactly like the encoding of the container presented earlier, except that a different Value Type is specified, a non-empty Text Value attribute is used, and the Continuity of Content attribute (which only applies to containers) is not included.

A text content item may have children, just like content items of any other value type, including containers.<sup>21</sup> Children are included by adding a Content Sequence:

```
(0040,a730) Content Sequence
... child content items go here ...
(fffe,e0dd) Sequence Delimitation Item
```

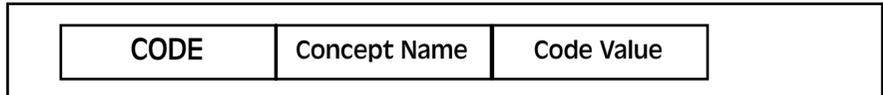
<sup>20</sup> It is forbidden as a consequence of the type of the Text Value (0040,A160) attribute being 1C rather than 2C. In traditional DICOM attributes, the meaning of a zero length value for a type 2 attribute is construed to be “unknown.” There is no such equivalent in SR, since a code value type with a coded value that means “unknown” should probably be used in these circumstances.

<sup>21</sup> This may be restricted by the SR SOP class in use.

## Value Type CODE

A value type of CODE contains a single coded entry value. The value is encoded as a sequence attribute, Concept Code Sequence (0040,A168),<sup>22</sup> with a single item which contains an instance of the code sequence macro (a “triplet” code). There may not be more than one item (since that would imply more than one value), and there may not be zero items (since that would imply no value).

Graphically, a CODE value type looks something like this:



As a short hand for describing a code value type, one can write:

```
<CODE:(,,“Anatomic Site”)=(,,“Breast”)>
```

Or, specifying the coded entries more precisely:

```
<CODE:(209104,99PMP,“Anatomic Site”=
(T-04000,SNM3,“Breast”)>
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type “CODE”
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value “209104”
(0008,0102) Coding Scheme Designator “99PMP”
(0008,0104) Code Meaning “Anatomic Site”
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value “T-04000”
(0008,0102) Coding Scheme Designator “SNM3”
(0008,0104) Code Meaning “Breast”
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

A coded entry content item may have children, just like content items of any other value type, including containers.<sup>23</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

22. The name of the attribute is “Concept Code Sequence” rather than something like “code value sequence”, since the same attribute is already used in the acquisition context module.

23. This may be restricted by the SR SOP class in use.

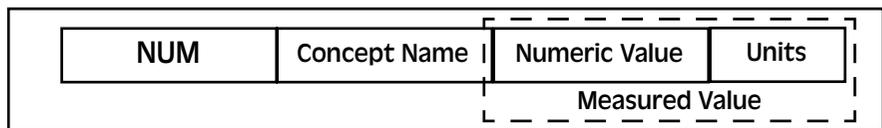
## Value Type NUM

A value type of NUM (numeric) contains a single numeric value, with units. It may contain no value at all, to convey the notion that the value is absent (unknown or missing).<sup>24</sup> The value is encoded as a sequence attribute, Measured Value Sequence (0040,A300), with zero or one item that contains two attributes:

- ▶ an attribute Numeric Value (0040,A30A), containing a single numeric value<sup>25</sup>
- ▶ a coded sequence attribute, Measurement Units Code Sequence (0040,08EA), with a single item to specify the units

Note that there is always a coded sequence for the units. It is not possible to encode a numeric value without units. If necessary, the units can be specified with a code that corresponds to a “dimensionless” number.<sup>26</sup>

Graphically, a NUM value type looks something like this:



As a short hand for describing a numeric value type, one can write something like this:

```
<NUM:(,,"Diameter")="1.3" (,,"centimeter")>
```

Or, specifying the coded entries more precisely:

```
<NUM:(209230,99PMP,"Diameter")=
      "1.3" (cm,UCUM,"centimeter")>
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type "NUM"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209230"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Diameter"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

24. This ability to specify a numeric content item with no value is inconsistent with all the other value types, which always require a value. The idea is to be able to specify a “fill in the blanks” style form that is not fully populated with numeric values, but contains concept names for what could be filled in. It remains to be seen how well supported this feature will be.

25. Even though the VR of the Numeric Value (0040,A30A) is specified in PS 3.6 with a value multiplicity (VM) of 1-n, for use in other objects, in the SR objects it is explicitly constrained to a VM of exactly 1.

26. This may, depending on the concept, be a code that means “arbitrary units,” or a code that means “unity.” In the former case the units are unspecified, in the latter case it means the number is dimensionless, i.e. as if the value was multiplied by “1.” See the Unified Code for Units of Measure (UCUM) for further discussion of this concept.

```

(0040,a300) Measured Value Sequence
(fffe,e000) Item
(0040,08ea) Measurement Units Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "cm"
(0008,0102) Coding Scheme Designator "UCUM"
(0008,0104) Code Meaning "centimeter"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a30a) Numeric Value "1.3"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
    
```

DICOM data sets are always ordered by ascending tag number. This includes the data set that is inside the measured value sequence item. As a side effect of the choice of attribute tag numbers, the measurement units code sequence is actually encountered before the numeric value itself.

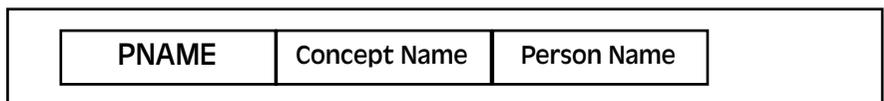
A numeric content item may have children, just like content items of any other value type, including containers.<sup>27</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

The numeric value itself is encoded as a Decimal String (DS) value representation. This means it may contain a signed or unsigned integer, a floating point number, or a number in scientific notation. Leading or trailing spaces are permitted, but not embedded spaces. The string has a maximum length of 16 bytes. It may not contain strings representing values such as positive or negative infinity, or "not-a-number" (NaN),<sup>28</sup> so care must be taken when creating the decimal string representation of a number stored internally as a binary floating point value.

## Value Type PNAME

A value type of PNAME contains a single value that is one person's name. The value is encoded as a DICOM Person Name (PN) value representation in the attribute Person Name (0040,A123).

Graphically, a PNAME value type looks something like this:



As a short hand for describing a person name value type, one can write:

```
<PNAME:(,,"Patient")="Smith^John">
```

Or, specifying the concept name more precisely:

27. This may be restricted by the SR SOP class in use.

28. See the IEEE 754:1985 floating point standard.

```
<PNAME:(209700,99PMP,"Patient")="Smith^John">
```

A dump of the actual DICOM attribute encoding, would look like:

```
(0040,a040) Value Type "PNAME"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209700"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Patient"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a123) Person Name "Smith^John"
```

In other words, this looks exactly like the encoding of the text value presented earlier, except that a different Value Type is specified and a non-empty Person Name attribute is used.

A person name content item may have children, just like content items of any other value type, including containers.<sup>29</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

The name itself is specified by separating individual components like last name and first name by the caret (^) symbol, and arranging them in a specified order. There may be up to five components, consisting of the family (last) name complex, the given (first) name complex, the middle name, the name prefix, and the name suffix.<sup>30</sup> One doesn't need to send all the components if the trailing components are empty, so there may be fewer than five components, and fewer than four '^' separators sent. For example, the following are all valid and equivalent ways to encode the name "John Smith":

<sup>29</sup> This may be restricted by the SR SOP class in use.

<sup>30</sup> This was the format proposed as the ANSI HISPP MSDS Person Name common data type when DICOM was being developed. Unfortunately, this form is not entirely consistent with HL7 2.x or ASTM, which have a separate component for the degree, as opposed to integrating this within the suffix component. Furthermore the "suffix" is inserted as a component before the prefix. That is, the HL7 components are "family name^given name^middle name^suffix^prefix^degree" as opposed to "family name^given name^middle name^prefix^suffix" in DICOM. The suffix is used in HL7 not to convey the degree, but other things like "Jr." or "III."

For example, in DICOM, the author's name and qualifications would be represented as "Clunie^David^A^Dr^MBBS,FRACR"

whereas in HL7 it would be

"Clunie^David^A^^Dr^MBBS,FRACR".

HL7 also allows for the concept of a "last name prefix" as a sub-component of the last name component, separated by a '&' delimiter. Hence, to use the example from HL7 2.3.1, "Ludwig van Beethoven" would be encoded as "Beethoven&van^Ludwig". There is no such convention is specified in DICOM.

HL7 also specifies an "extended patient name" data type which adds two further components, a type code to indicate whether the name is an alias, legal name, maiden name, etc., and a representation code to indicate whether it is an alphabetic, ideographic or phonetic representation.

The proposed HL7 V3 Data Types take a different approach altogether.

```
"Smith^John^^^"  
"Smith^John^"  
"Smith^John"
```

Each of the given and family name components may contain a more complex string than a simple name, and it may be formatted according to the named person's preference or local convention. For example, the following are all valid person name strings:

```
"Hamilton-Smythe^John"  
"Hamilton Smythe^John"  
"hamilton-smythe^john"  
"HAMILTON-SMYTHE^J."
```

These are probably all intended to refer to the same individual, even though case sensitive literal string matching would not make the equivalence apparent.<sup>31</sup> There is also the question of what constitutes a "given name" as opposed to a middle name,<sup>32</sup> and hence these two forms are both valid and probably equivalent:

```
"Smith^John^Thomas"  
"Smith^John Thomas"
```

To make things worse, there is a historical precedent for ignoring the caret separators altogether. It is valid (though very unhelpful) to send:

```
"John Smith"  
"Smith, John"  
"Smith, J."  
"J. Smith"
```

and so on. This was originally permitted in DICOM to allow support for legacy systems that did not distinguish name components in their user interface, database or internal header fields. Fortunately, the use of the "proper" caret-separated components is increasing, as integration with information systems becomes a higher priority. Regardless, it is strongly recommend that structured reports make use of properly "componentized" names.

The use of the suffix and prefix component is fairly straightforward in English speaking regions. A personal or professional title like "Ms." or "Dr." or "Sir" go in the prefix. Academic degrees and the courtesy title "Esquire" go in the suffix. For example:

```
"Smith^John^^Dr.^MD,PhD"  
"Smith^John^^^Esq."  
"Smith^John^^Sir"
```

Whether or not medical practitioners are addressed by personal title or professional title varies from country to country. There is further variation in non-English speaking countries. For example, there may be additional titles corresponding to the person's

---

31. By the way, a recent change to the DICOM Standard allows for case insensitive matching of Person Name (PN) values during queries, to alleviate problems like this. See CP 190.

32. Note that the standard refers to given name and family name "complexes" for this reason.

profession, such as “Bestruurder” (“Director”) in the Netherlands, “Avvocato” or “Avvocatessa” (“Lawyer”) in Italy and “Ingenieur” (“Engineer”) in Germany.<sup>33</sup>

Further internationalization issues arise in countries where the language has a phonetic or ideographic representation, such as in Japan and Korea. For these situations, DICOM allows up to three “component groups,” the first a single-byte representation as is used for western languages, then an ideographic (Kanji or Hanga) representation and then a phonetic representation (Hiragana or Hangul). These are separated by ‘=’ (0x3d) characters. ISO 2022 escape sequences (code extension techniques) may be used to change character sets, depending on the choices specified in the Specific Character Set attribute.<sup>34</sup> Each of the component groups is limited to 64 characters.<sup>35</sup> A typical Japanese PNAME looks like this:

“Miyamoto^Musashi= 宮本 ^ 武蔵 = みやもと ^ むさし ”

Finally, there are a number of other important characteristics of the name that is contained in the value:

- ▶ it contains only a single value, so the backslash ('\') character is *not* permitted
- ▶ it may contain any character in the “default character repertoire” or as defined by the Specific Character Set (0008,0005)
- ▶ it may be padded with trailing spaces, which may be ignored
- ▶ leading spaces are considered to be significant (whatever that means in the context of a name); specifically, they may not be elided when stored

When a PNAME value type content item is used:

- ▶ it *must* have a concept name, and
- ▶ it *must* have a value with a length of one or more characters.

One cannot convey the idea of a blank name or placeholder or unknown value by a zero length name value, which is explicitly forbidden.<sup>36</sup>

## Value Types DATE, TIME and DATETIME

A value of type DATE, TIME or DATETIME contains a single value that represents, respectively, either:

- ▶ a date that is a whole day (without a time)

---

33. These examples are taken from De Vries M. *Internationally Yours*. Houghton Mifflin 1994. ISBN 0-395-69611-9.

34. See the chapter on “Internationalization and Localization” for more details.

35. Sixty-four characters are specified per component group, rather than 64 bytes, since in some character sets, a character may occupy more than one byte. The total length of all the component groups may be three times 64 plus two ‘=’ delimiter characters, that is 194 characters.

36. It is forbidden as a consequence of the type of the Person Name (0040,A123) attribute being 1C rather than 2C. In traditional DICOM attributes, the meaning of a zero length value for a type 2 attribute is construed to be “unknown.” There is no such equivalent in SR, since a code value type with a coded value that means “unknown” should probably be used in these circumstances.

- ▶ a time on any day (without a date)
- ▶ a time on a particular date

These value types correspond to the traditional DICOM string value representations DA, TM and DT. Indeed, those value representations are used to encode the SR value types, as follows:

- ▶ attribute Date (0040,A121) is encoded with a DA VR
- ▶ attribute Time (0040,A122) is encoded with a TM VR
- ▶ attribute DateTime (0040,A120) is encoded with a DT VR

There may not be more than one value, and the value may not be zero length.

Graphically, DATE, TIME and DATETIME value types look something like this:

DATE	Concept Name	DA Value
TIME	Concept Name	TM Value
DATETIME	Concept Name	DT Value

As a short hand for describing a date-time value type, one can write:

```
<DATETIME:(,,“Admitted”)=(“200006100913”)>
```

Or, specifying the coded entries more precisely:

```
<DATETIME:(209701,99PMP,“Admitted”)="200006100913">
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type "DATETIME"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209701"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Admitted"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a120) DateTime "200006100913"
```

A date or time entry content item may have children, just like content items of any other value type, including containers.<sup>37</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

The rules for constructing the date and time strings for all three value types and representations are similar:

<sup>37</sup>. This may be restricted by the SR SOP class in use.

- ▶ each string consists of components of increasing “granularity” from left to right, i.e. the year precedes the month which precedes the day, etc.
- ▶ any trailing component, if not used, can be omitted
- ▶ the date is specified by year, month and day
- ▶ the year is always four digits, never two (i.e. “1997”, not “97”)
- ▶ the month is always two digits and never letters (i.e. “03”, not “3” or “MAR”)
- ▶ the time is specified in hours (of a 24 hour clock), minutes and seconds and fractional seconds
- ▶ fractional seconds may be specified to a precision of up to 6 digits<sup>38</sup>
- ▶ no delimiter is specified between components of the date, components of the time, or between the date and time, except for the period ‘.’ between the seconds and fractional seconds, which must be absent if fractional seconds are not specified<sup>39</sup>

For example:

- ▶ the 10th of June, 2000 would be encoded as:  
“20000610”
- ▶ 4 pm could be encoded as any of:  
“16”  
“1600”  
“160000”  
“160000.0”  
“160000.000000”
- ▶ 45 mins, 13.691 secs past 1 pm on the 10th of June, 2000 could be encoded as:  
“20000610134513.691”

How does one determine in which time zone the time alone, or the time component of a date-time, is specified? The DT value representation allows a time zone suffix to be explicitly added, in the form of a 5 byte suffix. It consists of either a plus (+) or minus (-), followed by a two digit hour and two digit minute offset from Universal Coordinated Time (UTC).<sup>40</sup> Exactly 5 bytes must be present; the plus sign may not be omitted.

For example, 1pm on the 10th of June, 2000 in Paris (Middle European Time) in the winter (i.e. without daylight saving) could be written:

“200006101300+0100”

---

38. The total length of all three VRs is constrained, to 8, 16 and 26 bytes for DA, TM and DT, respectively.

39. For historical reasons, one is also allowed to accept (but not create) date components in a DA VR separated by periods, e.g. “2000.06.10”, and time components in a TM VR separated by a colon ‘:’, e.g. “16:00:00”. This is not permitted for the DT VR, however, since there are no backward compatibility issues.

40. Formerly known as Greenwich Mean Time (GMT). Where the order of the letters in the “UTC” abbreviation come from is unknown to the author.

In this example, the trailing seconds component has been omitted before the time zone suffix.

Historically, there was no way to specify the time zone for TM attributes. This has recently changed.<sup>41</sup> The SOP Common Module now contains an optional Timezone Offset From UTC (0008,0201). The value in this attribute is a five byte string encoded in exactly the same way as the time zone suffix in the DT attribute. For example:

```
(0008,0201) Timezone Offset From UTC "+0100"
```

When this attribute is present, it applies to *all* the TM attributes in the object. It does not apply to *any* of the DT attributes, each of which must have the time zone explicitly specified, if required. Note that the only way to have different time zones for different attributes in the same object is by using DT attributes.

When the time zone is not specified explicitly, either in a DT attribute or in Timezone Offset From UTC (0008,0201), then the time zone is undefined. In practice, it will almost always be the local time zone in which the object was created (with any applicable daylight saving correction), but it is not required to be so. It is always safest to be explicit, and specify the time zone in all DT attributes as well as in Timezone Offset From UTC (0008,0201).

## Reference Value Types

There are four different, but related, reference value types:<sup>42</sup>

- ▶ references to image objects ("IMAGE")
- ▶ references to waveforms ("WAVEFORM")
- ▶ references to other composite objects, such as reports ("COMPOSITE")
- ▶ references to DICOM Unique Identifiers ("UIDREF")

Figure 5 illustrates the appropriate choice of reference value type, based on the type of object or entity that is referenced.

A composite object is an object that contains information about multiple entities in the DICOM information model. Specifically, a composite object combines in a single object information about the patient, study, series and instance. Images, waveforms and structured reports are all composite objects. The DICOM storage and query-retrieve services are used to "manage" composite objects.

---

41. The time zone offset attribute was added in CP 131.

42. Note that references to DICOM objects in an SR document make use of their unique identifiers (UIDs). As such, they identify "what" is being referenced, but do not specify "where" the referenced object might be. How to actually get hold of the referenced object is outside the scope of the structure of the SR document. There are "hints" provided in the management information, such as the Retrieve AE Title, but as will be discussed later in the chapter on "Document Management: SR in the Real World," this information may become stale over time, and should not be relied upon.

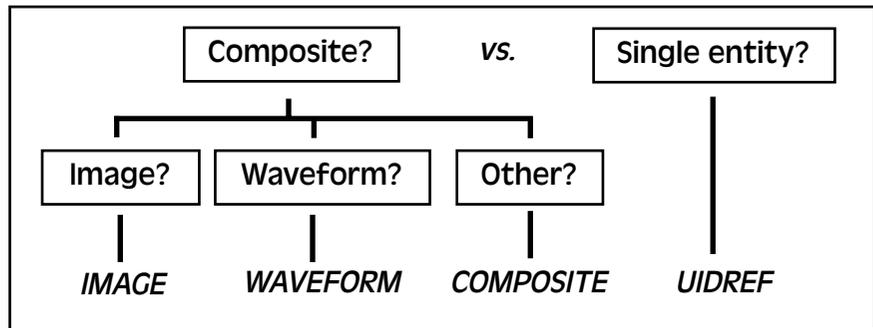


FIGURE 5. Reference Value Types

The IMAGE and WAVEFORM value types are extensions of the COMPOSITE value type that allow references to sub-components of an image or waveform, specifically to frames and waveform channels. Image and waveform value types are also allowed to be the targets of SELECTED FROM relationships for spatial and temporal coordinate value types (SCoord and TCoord).

The COMPOSITE value type is never used for images or waveforms. It may be used for any other composite object, including:

- ▶ other structured reports
- ▶ standalone overlays and curves<sup>43</sup>
- ▶ radiotherapy objects that are not images<sup>44</sup>

For example, from the current document, one might want to reference a prior report for a previous study that was stored in a DICOM SR object. The COMPOSITE value type would be appropriate for this. In contrast, to reference an image that was acquired as part of a previous study one would use the IMAGE value type.

The UIDREF value type is used to reference non-composite objects, such as previous studies themselves, as opposed to reports about them or images within them. The value contains an identifier of an individual “normalized” object that represents a single entity in the DICOM information model, such as a patient, study, study component or series.

For example, from the current document, one might want to reference a previous study itself. Such a study is either identified “abstractly” by a Study Instance UID, or by the SOP Instance UID of a normalized object.<sup>45</sup> The UIDREF value type would be appropriate for this.

The referenced entity in the UIDREF value is either:

43. Neither of which is in widespread use.

44. There is also a radiotherapy image object that would be referenced using an IMAGE value type.

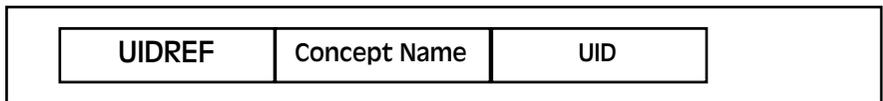
45. Specifically, an instance of the Detached Study Management SOP class.

- ▶ a concrete instance of an individual entity, stored in a normalized object, identified by the SOP Instance UID of the normalized object
- ▶ an abstract entity, described consistently in all composite objects, identified by a specific attribute like Study Instance UID, Series Instance UID, etc.

## Value Type UIDREF

A value type of UIDREF (unique identifier reference) contains a single string value that contains a DICOM UID. The value is encoded as a DICOM UID (UI) value representation in the attribute UID (0040,A124).

Graphically, a UIDREF value type looks something like this:



As a short hand for describing a UID reference value type, one can write:

```
<UIDREF:(,,"Study Instance UID")="1.20.51.96.22.134678">
```

Or, specifying the concept name more precisely:

```
<UIDREF:(209702,99PMP,"Study Instance UID")="...">
```

A dump of the actual DICOM attribute encoding, would look like:

```
(0040,a040) Value Type "UIDREF"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209702"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Study Instance UID"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a124) UID "1.20.51.96.22.134678"
```

In other words, this looks exactly like the encoding of the text value presented earlier, except that a different Value Type is specified and a non-empty UID attribute is used.

A UIDREF content item may have children, just like content items of any other value type, including containers.<sup>46</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

The UID itself is specified as a string of numeric values ("components") separated by periods.

There are a number of other important characteristics of the UID that is contained in the value:

<sup>46</sup>. This may be restricted by the SR SOP class in use.

- ▶ it contains only a single value, so the backslash ‘\’ character is *not* permitted
- ▶ it may be not be padded with spaces
- ▶ odd length UIDs are padded with a zero byte (0x00), *not* with a space, and *not* with the digit ‘0’ (0x30)<sup>47</sup>
- ▶ leading zeroes are prohibited in components, so for example “1.02.1” is illegal, and should be written “1.2.1”<sup>48</sup>
- ▶ a component may have a value of zero, so “1.0.1” is legal
- ▶ it may not exceed 64 characters (bytes) in length<sup>49</sup>

When a UIDREF value type content item is used:

- ▶ it *must* have a concept name, which specifies the purpose of the reference to the object, and
- ▶ it *must* have a value with a length of one or more characters.

One cannot convey the idea of a blank UID or placeholder or unknown value by a zero length UID value, which is explicitly forbidden.<sup>50</sup>

Unique identifier references are used to point to DICOM entities, such as studies, study components, procedure steps and so on. They are *never* used to point to composite objects, including other reports, images and waveforms, which have more specific reference mechanisms.

## Value Type COMPOSITE

A value type of COMPOSITE (composite object reference) contains a single reference to a DICOM composite object that is not an image or a waveform. The value is encoded in a sequence, Referenced SOP Sequence (0008,1199), with a single item which contains the SOP Class UID and SOP Instance UID of the referenced object.<sup>51</sup> Note that there may not be more than one item (since that would imply more than one value),<sup>52</sup> and there may not be zero items (since that would imply no value).

---

47. The digit ‘0’ (0x30) is mentioned, because one popular but flawed implementation once did this. Obviously ‘0’ can’t be used because it would then masquerade as an even length UID with an extra zero on the last numeric component. For example “1.2” would masquerade as “1.20”.

48. Conceptually these two values are equivalent, but only the canonical non-zero padded encoding is permitted in DICOM. This considerably simplifies string matching of UIDs.

49. In this case, the number of characters does always equal the number of bytes, since the Specific Character Set does not apply to the UI value representation. That is, the default character repertoire of ISO 646 (US-ASCII) is always used for encoding unique identifiers.

50. It is forbidden as a consequence of the type of the UID (0040,A124) attribute being 1C rather than 2C. In traditional DICOM attributes, the meaning of a zero length value for a type 2 attribute is construed to be “unknown.” There is no such equivalent in SR, since a code value type with a coded value that means “unknown” should probably be used in these circumstances.

51. The SOP Instance UID uniquely identifies the instance of the referenced object, while the SOP Class UID identifies what type (class) of object it is.

Graphically, a COMPOSITE value type looks something like this:



As a short hand for describing a composite value type, one can write:<sup>53</sup>

```
<COMPOSITE:(,,“Prior Report”)=(Basic Text SR,“1.2.3.4”)>
```

Or, specifying the coded entries more precisely:

```
<COMPOSITE:(209021,99PMP,“Prior Report”
=(Basic Text SR,“1.2.3.4”)>
```

A dump of the actual DICOM attribute encoding, would look as follows:<sup>54</sup>

```
(0008,1199) Referenced SOP Sequence
(fffe,e000) Item
(0008,1150) Referenced SOP Class UID
                “1.2.840.10008.5.1.4.1.1.88.11”
(0008,1155) Referenced SOP Instance UID “1.2.3.4”
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a040) Value Type “COMPOSITE”
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value “209021”
(0008,0102) Coding Scheme Designator “99PMP”
(0008,0104) Code Meaning “Prior Report”
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

A COMPOSITE content item may have children, just like content items of any other value type, including containers.<sup>55</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

When a COMPOSITE value type content item is used it may have a concept name, which specifies the purpose of the reference to the object.

Composite references are used to point to composite objects such as other reports, that are not images or waveforms. They are *never* used to point to individual entities

52. This is not strictly true. As of the time of writing, there is what is probably an error in Supplement 23 that has not yet been corrected with a CP. Though the text describing the COMPOSITE value type implies a single reference, the macro allows for one or more items in the sequence. This also affects the IMAGE and WAVEFORM value types which include the COMPOSITE macro.

53. In this shorthand, the SOP class is referred to by its name, Basic Text SR (Storage), rather than by its UID which is “1.2.840.10008.5.1.4.1.1.88.11”.

54. The Referenced SOP Sequence attribute actually appears before the Value Type in the encoding, as an accident of the assignment of the group and element tags, which are sorted in ascending order by tag in the bit stream.

55. This may be restricted by the SR SOP class in use.

such as normalized objects or abstract identifiers of individual entities, such as studies or series; in these cases the UIDREF value type is always used.

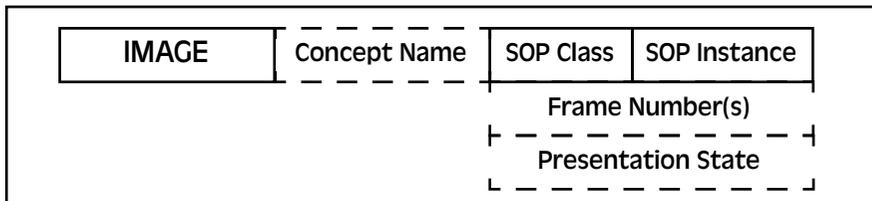
## Value Type IMAGE

A value type of IMAGE (image object reference) contains a single reference to a DICOM image object. The encoding of the value type is an extension of the COMPOSITE value type, which contains a value that is encoded in a sequence, Referenced SOP Sequence (0008,1199), with a single item which contains the SOP Class UID and SOP Instance UID of the referenced object.<sup>56</sup> In the case of the IMAGE value type, there also may be:

- ▶ a Referenced Frame Number (0008,1160), which identifies one or more frames as being referenced, if the reference is to a multi-frame image; if this attribute is not included in a reference to a multi-frame image, then all the frames are referenced
- ▶ an optional reference to a Grayscale Softcopy Presentation State Storage object,<sup>57</sup> encoded in a nested Referenced SOP Sequence containing exactly one item

Note that there may not be more than one item (since that would imply more than one value),<sup>58</sup> and there may not be zero items (since that would imply no value).

Graphically, an IMAGE value type looks something like this:



As a short hand for describing an image value type, one can write:<sup>59</sup>

```
<IMAGE:(,, "End Systole")=(XA Image, "1.2.3.4", 34, Masked)>
```

Or, specifying the coded entries more precisely:

56. The SOP Instance UID uniquely identifies the instance of the referenced object, while the SOP Class UID identifies what type (class) of object it is.

57. Grayscale Softcopy Presentation State Storage objects are used to specify appropriate spatial and contrast transformations to apply to an image to display or print it with a certain appearance. They may also contain graphic annotation information. For example, such an object may specify the appropriate window center and width for a CT image object. The specification of a standardized grayscale space in the presentation state also helps achieve consistency of appearance between different monitors and printers. For multi-frame objects, the presentation state may also specify how to mask (subtract) frames. There is no equivalent mechanism for color images. See the chapter on "Images, Waveforms, and Voice Audio" for further details.

58. This is not strictly true. As of the time of writing, there is what is probably an error in Supplement 23 that has not yet been corrected with a CP. Though the text describing the COMPOSITE value type implies a single reference, the macro allows for one or more items in the sequence. This also affects the IMAGE and WAVEFORM value types which include the COMPOSITE macro.

```
<IMAGE:(209801,99PMP,"End Systole")=
      (XA Image,"1.2.3.4",34,Masked)>
```

A dump of the actual DICOM attribute encoding, would look as follows:<sup>60</sup>

```
(0008,1199) Referenced SOP Sequence
(fffe,e000) Item
(0008,1150) Referenced SOP Class UID
      "1.2.840.10008.5.1.4.1.1.12.1"
(0008,1155) Referenced SOP Instance UID "1.2.3.4"
(0008,1160) Referenced Frame Number "34"
(0008,1199) Referenced SOP Sequence
(fffe,e000) Item
(0008,1150) Referenced SOP Class UID
      "1.2.840.10008.5.1.4.1.1.11.1"
(0008,1155) Referenced SOP Instance UID "1.2.3.5"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a040) Value Type "IMAGE"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209801"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "End Systole"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

An IMAGE content item may have children, just like content items of any other value type, including containers.<sup>61</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

When an IMAGE value type content item is used it may have a concept name, which specifies the purpose of the reference to the image.

Image references are used only to point to images, never other composite objects that are not images, such as waveforms or other reports. They are never used to point

---

59. In this shorthand, the SOP class is referred to by its name, XA Image (Storage), rather than by its UID which is "1.2.840.10008.5.1.4.1.1.12.1". Also, the presentation state is referred to by its name, such as might be contained in the Presentation Label (0070,0080) attribute in the presentation object, rather than its SOP Instance UID. The SOP Class UID of the presentation state ("1.2.840.10008.5.1.4.1.1.11.1") has also been elided.

60. Note the nesting of the Referenced SOP Sequence attributes, the outer of which is the reference to the image itself, and the inner of which is the reference to the presentation state. Careful note should be taken of the use of nested macros in the standard to describe this pattern. In particular, the image reference macro adds attributes to the Referenced SOP Sequence whose definition begins in the composite reference macro - this is indicated by the nesting level ">" that precedes the Referenced Frame Number and the presentation state Referenced SOP Sequence.

61. This may be restricted by the SR SOP class in use.

to individual entities such as normalized objects or abstract identifiers of individual entities, such as studies or series; in these cases the UIDREF value type is always used.

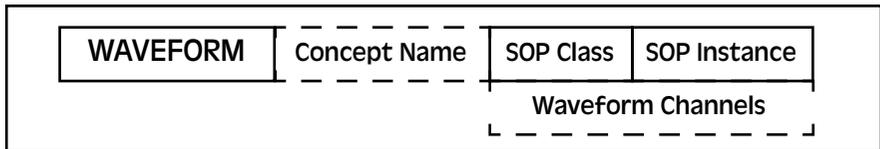
## Value Type WAVEFORM

A value type of WAVEFORM (waveform object reference) contains a single reference to a DICOM waveform object. The encoding of the value type is an extension of the COMPOSITE value type, which contains a value that is encoded in a sequence, Referenced SOP Sequence (0008,1199), with a single item which contains the SOP Class UID and SOP Instance UID of the referenced object.<sup>62</sup> In the case of the WAVEFORM value type, there is also a Referenced Waveform Channels (0040,A0B0) attribute, which identifies all or one or more channels within multiplex groups in the waveform, if all the channels in all the multiplex groups are not included in the reference.<sup>63</sup>

Waveforms are structured in blocks of samples, called multiplex groups, each containing multiple channels. In a WAVEFORM reference, channels are always referenced as pairs (M,C), that is, multiplex group and channel. The multiplex groups and channels are numbered from one (not zero). If the channel number is zero, then all the channels of a multiplex group are included in the reference.<sup>64</sup> The temporal coordinate mechanism (TCOORD value type) may be used to further specify a point or segment of a channel reference, by absolute or relative time, or by sample number.

Note that there may not be more than one item (since that would imply more than one value),<sup>65</sup> and there may not be zero items (since that would imply no value).

Graphically, a WAVEFORM value type looks something like this:



As a short hand for describing a waveform value type, one can write something like this, corresponding to the choice of all the channels in multiplex group two (Leads aVR, aVL and aVF):<sup>66</sup>

```
<WAVEFORM:(,, "aV Leads")=(12-Lead ECG,"1.2.3.4","2,0")>
```

62. The SOP Instance UID uniquely identifies instance of the referenced object, while the SOP Class UID identifies what type (class) of object it is.

63. There is not (yet) any equivalent of the image presentation state object for waveforms.

64. The Referenced Waveform Channels (0040,A0B0) attribute is encoded as an unsigned 16-bit binary value (a VR of US) with a multiplicity of 2-2n.

65. This is not strictly true. As of the time of writing, there is what is probably an error in Supplement 23 that has not yet been corrected with a CP. Though the text describing the COMPOSITE value type implies a single reference, the macro allows for one or more items in the sequence. This also affects the IMAGE and WAVEFORM value types which include the COMPOSITE macro.

66. In this shorthand, the SOP class is referred to by its name, 12-Lead ECG Waveform (Storage), rather than by its UID which is "1.2.840.10008.5.1.4.1.1.9.1.1".

Or, specifying the coded entries more precisely:

```
<WAVEFORM:(209802,99PMP,"aV Leads")=
(12-Lead ECG,"1.2.3.4","2,0")>
```

A dump of the actual DICOM attribute encoding, would look as follows:<sup>67</sup>

```
(0008,1199) Referenced SOP Sequence
(fffe,e000) Item
(0008,1150) Referenced SOP Class UID
"1.2.840.10008.5.1.4.1.1.9.1.1"
(0008,1155) Referenced SOP Instance UID "1.2.3.4"
(0040,a0b0) Referenced Waveform Channels "2,0"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a040) Value Type "WAVEFORM"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209802"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "aV Leads"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

A WAVEFORM content item may have children, just like content items of any other value type, including containers.<sup>68</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

When a WAVEFORM value type content item is used it may have a concept name, which specifies the purpose of the reference to the waveform.

Waveform references are used only to point to waveforms, never other composite objects that are not waveforms, such as images or other reports. They are *never* used to point to individual entities such as normalized objects or abstract identifiers of individual entities, such as studies or series; in these cases the UIDREF value type is always used.

## Coordinate Value Types

There are two value types that may be used to reference a "subset" of an image or waveform:

- ▶ spatial coordinates
- ▶ temporal coordinates

---

<sup>67</sup>. Note the nesting of the Referenced Waveform Channels attribute. Careful note should be taken of the use of nested macros in the standard to describe this pattern. In particular, the waveform reference macro adds attributes to the Referenced SOP Sequence whose definition begins in the composite reference macro - this is indicated by the nesting level ">" that precedes the Referenced Waveform Channels.

<sup>68</sup>. This may be restricted by the SR SOP class in use.

Spatial coordinates can only be applied to images, and not waveforms, since waveforms do not have two spatial dimensions. Temporal coordinates can be used to select points or periods of time from either (or both) waveforms *and* multi-frame images.

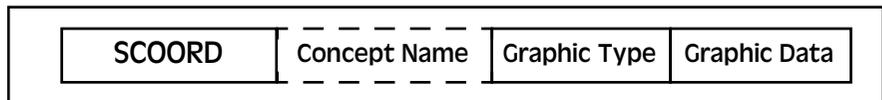
The references to the images or waveforms themselves are encoded as separate content items of value type IMAGE or WAVEFORM, not within the coordinate value types. The object reference and the coordinate content items are related by use of the SELECTED FROM relationship, as described in later chapters.<sup>69</sup> Sufficeth to say for now, the use of separate coordinate and object reference content items allows for:

- ▶ a single spatial or temporal coordinate set to apply to multiple images and/or waveforms, for example, the point of end diastole on an angiographic image and a synchronously acquired physiological waveform
- ▶ selection of both temporal and spatial coordinates from an image or images, for example, the outline of the left ventricular volume on a frame at end diastole

## Value Type SCOORD

A value type of SCOORD (spatial coordinates) contains a single set of spatial coordinates. The encoding of the value type is as a set of points in Graphic Data (0070,0022) representing the x and y points of a geometric object of the form specified in Graphic Type (0070,0023).<sup>70</sup>

Graphically, an SCOORD value type looks something like this:



As a short hand for describing a spatial coordinate value type, one can write:

```
<SCOORD:(,,"Epicenter")=(POINT,128.5,128.5)>
```

Or, specifying the coded entries more precisely:

```
<SCOORD:(209239,99PMP,"Epicenter")=(POINT,128.5,128.5)>
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type "SCOORD"  
(0040,a043) Concept Name Code Sequence  
(ffff,e000) Item
```

69. Specifically, the SELECTED FROM relationship is introduced in the chapter on "Encoding & Relationships: Building the Tree," and further explained in the chapter on "Images, Waveforms, and Voice Audio."

70. This means of describing vector graphic objects is very similar to the Graphic Annotation Module of the Grayscale Softcopy Presentation State Storage object. The differences are that in SR a POLYLINE is always closed, there is a MULTIPOINT graphic type, and the INTERPOLATED type does not apply.

```
(0008,0100) Code Value "209239"  
(0008,0102) Coding Scheme Designator "99PMP"  
(0008,0104) Code Meaning "Epicenter"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0070,0022) Graphic Data "128.5,128.5"  
(0070,0023) Graphic Type "POINT"
```

An SCOORD content item will *always* have one or more children that are of IMAGE value type.<sup>71</sup> It is not possible to use an SCOORD to refer to an arbitrary point in space outside the context of an image, as discussed later. Children are included by adding a Content Sequence, as shown previously for the text value type.

When an SCOORD value type content item is used it may have a concept name, which specifies the purpose of the reference to the spatial coordinates.

The choice of graphic types available is:

- ▶ POINT, which specifies a single point
- ▶ MULTIPOINT, which specifies multiple, independent points
- ▶ POLYLINE, which specifies a closed polygon defined by its vertices
- ▶ CIRCLE, which specifies a circle defined by its center and a point on the perimeter
- ▶ ELLIPSE, which specifies an ellipse defined by the endpoints of its major and minor axes

The graphic data attribute consists of pairs of points, the first of the pair being the X or column location, and the second being the Y or row location. These are encoded as floating point values<sup>72</sup> with sub-pixel resolution. This means that:

- ▶ the top left hand corner of the top left hand pixel is 0.0,0.0
- ▶ the center of the top left hand pixel is 0.5,0.5
- ▶ the bottom right hand corner of the top left hand pixel is 1.0,1.0

Further more, the coordinates are not normalized to a particular range like 0.0 to 1.0, but rather are specified in units of actual pixels in the image. So, for example, given an image with 512 columns and 256 rows:

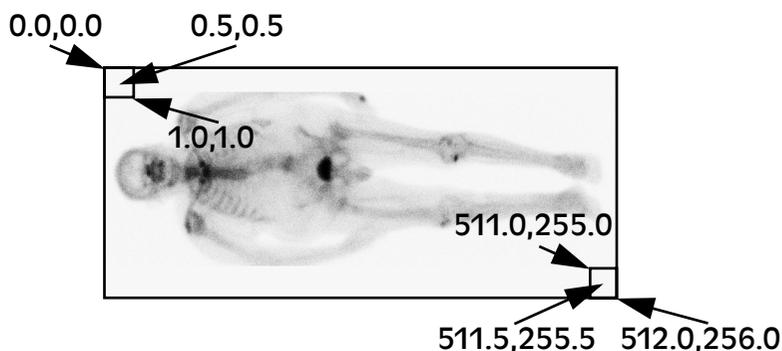
- ▶ the top left hand corner of the bottom right hand pixel is 511.0,255.0
- ▶ the center of the bottom right hand pixel is 511.5,255.5
- ▶ the bottom right hand corner of the bottom right hand pixel is 512.0,256.0

---

71. The IMAGE children will also always have a SELECTED FROM relationship with the SCOORD parent, as described later.

72. These really are IEEE 754:1985 32 bit binary floating point values with a DICOM VR of Float (FL), and not decimal strings.

This is illustrated in Figure 6.



**FIGURE 6. Spatial Coordinates With Sub-Pixel Resolution**

Since the coordinates are image specific, rather than normalized, they obviously having no meaning unless applied to a specific image.<sup>73</sup> Furthermore, there can be no change in geometry of the referenced image, otherwise the coordinates would be invalidated.<sup>74</sup>

It is not currently possible to specify (in a structured report) an arbitrary location or region in three dimensions relative to the patient, independently of an image. Given an image-relative set of coordinates, if the referenced image has patient-relative spatial location information (such as is the case for CT and MR images), the two-dimensional image-relative coordinates can be mapped into patient-relative three-dimensional coordinates.<sup>75</sup> Without indirection “through an image,” however, no three-dimensional points may be specified. The radiotherapy family of objects does have a mechanism to specify image-independent three-dimensional locations,<sup>76</sup> and this might perhaps be added to SR in the future.

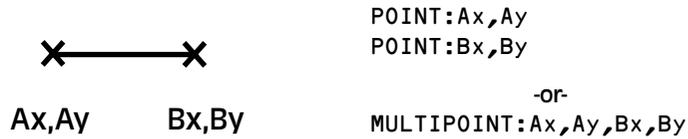
The choice of graphic type to use depends on the application:

73. As an aside, the Grayscale Softcopy Presentation State object defines coordinates in an “image-relative” space for graphics that are applied prior to any spatial transformation or displayed area selection, and in a “displayed-area-relative” space for graphics that are applied afterwards. However, since the characteristics of the actual display cannot be known a priori, “displayed-area-relative” coordinates are specified in a normalized units (from 0.0 to 1.0). The SR spatial coordinate value type uses only the “image-relative” space.

74. One may well ask why anyone would want to change the size of an image without creating a new SOP Instance. Strictly speaking, the standard requires that a new SOP Instance UID must be created only if the “interpretation” of the image is changed. Stranger things have happened, so be wary.

75. Specifically, by use of the attributes of the Image Plane Module, Image Position (Patient) (0020,0032), Image Orientation (Patient) (0020,0037) and Pixel Spacing (0028,0030), together with a little high school geometry.

76. Defined in IEC 1217, “Radiotherapy Equipment - Coordinates, Movements and Scales.”



**FIGURE 7. Spatial Coordinates for Distance Along a Line**

- ▶ the POINT graphic type may be used to indicate a single location of significance, such as the epicenter of a mass
- ▶ the MULTIPOINT graphic type may be used to indicate the locations of the epicenters of clustered entities, such as a cluster of micro-calcifications on a mammogram; alternatively a closed POLYLINE, a CIRCLE or an ELLIPSE could be specified that enclosed such a cluster
- ▶ indicating the end-points of a distance measurement along a straight line can be achieved in several ways; two separate SCOORD content items could be defined, one for each end-point, each with a purpose of reference of “end-point,” with their common parent indicating the type measurement and its numeric value; alternatively, a single MULTIPOINT content item could be used, with two points and a purpose of reference of “end-points” or similar; a simple extension would be to combine the length of multiple segments in a MULTIPOINT by using more than two points<sup>77</sup> (see Figure 7)
- ▶ outlines of objects for the purpose of indicating the source of circumferential linear distance, area, volume or other quantitative measures like mean density,<sup>78</sup> may be indicated by using the CIRCLE, ELLIPSE or POLYLINE value types; in the case of a POLYLINE, there need to be more than two points; the standard does not address whether or not the start and end points of a POLYLINE should be the same to indicate that the object is “explicitly closed,” or whether this closure is implicit since a POLYLINE is by definition always closed (i.e. are four or five points necessary to specify a rectangle)<sup>79</sup> (see Figure 8)
- ▶ an angle between two lines (protractor) may be indicated in a number of ways, such as by using single POINT content items with appropriate purposes of reference to indicate the ends of the lines and the common origin or vertex, or by a single MULTIPOINT of three points whose middle point is the vertex, or by two MULTIPOINT content items with a common point that is the vertex (see Figure 9)

77. A POLYLINE cannot be used because in SR it is defined to always represent a closed polygon.

78. Such as mean Hounsfield Units (HU) on a CT image.

79. In a presentation state, five points are necessary to define a rectangle using a POLYLINE, since there is no implicit closure, and only objects with the same start and end point may be closed.

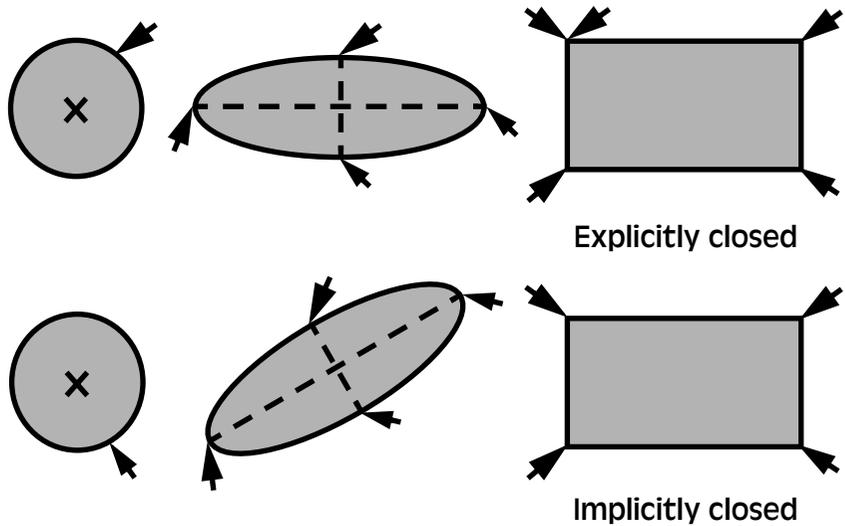


FIGURE 8. Spatial Coordinates for Closed Objects

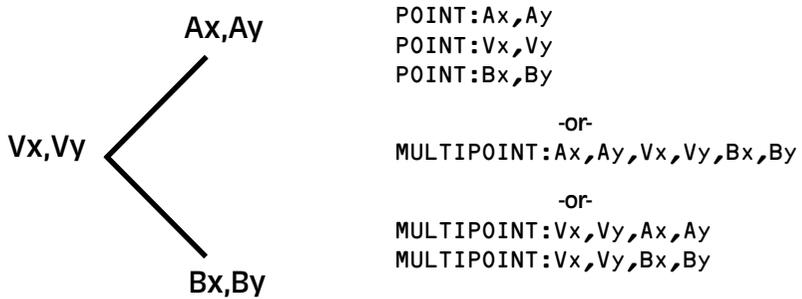
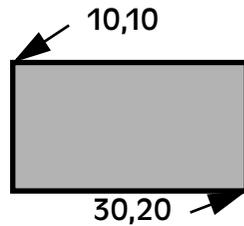


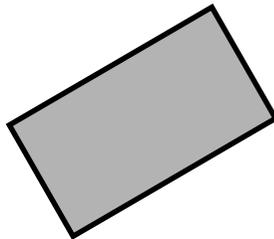
FIGURE 9. Spatial Coordinates for Angles Between Lines

- ▶ a special case of a POLYLINE is the rectangle, which obviously has certain constraints on the relationship between its vertices; though rectangles are very commonly used, there is no special graphic type for them, since defining such would require definition of two vertices as well as an angle (since rectangles may be rotated); it remains for the receiver of a POLYLINE object to recognize patterns of closed polylines that are consistent with rectangles if necessary<sup>80</sup> (see Figure 10)

80. For instance, in order to render them on a display or printer using optimized graphics primitives that draw rectangles in a more specific manner than arbitrary polygons.



POLYLINE: 10,10,  
30,10,  
30,20,  
10,20,  
10,10



POLYLINE: 10,10,  
27.3,20,  
32.3,6.3,  
15,1.3,  
10,10

Rotated 30° CCW

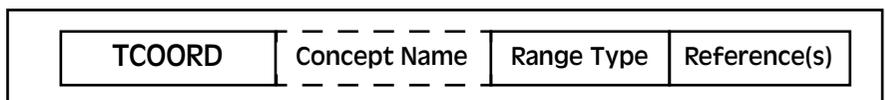
FIGURE 10. Spatial Coordinates for Rectangles Including Rotation

From the foregoing, one might rightfully be concerned that the generality of the graphic types available in SR creates some risk for interoperability. Hopefully, templates will be developed to describe common measurement patterns. These could restrict the choice of graphic type for a particular measurement (as well as define the codes to describe the pattern and to be used for measurement units). Such templates are needed not just for measurements, but also to define consistent ways in which to localize and identify specific image features, such as the epicenter or outline of a mass, and so on. This subject is addressed further in the chapter on “Templates: Bringing order to chaos ...”.

### Value Type TCOORD

A value type of TCOORD (temporal coordinates) contains one or more sets of temporal coordinates. The encoding of the value type is as a set of sample offsets, time offsets or dates and times, representing one or more points or ranges in time of the form specified in Temporal Range Type (0040,A130).<sup>81</sup>

Graphically, a TCOORD value type looks something like this:



<sup>81</sup> This means of describing temporal coordinates is exactly the same as in the Waveform Annotation Module of the Waveform objects.

As a short hand for describing a temporal coordinate value type, one can write:

```
<TCOORD:(,,"P wave")=(POINT,"0.735")>
```

Or, specifying the coded entries more precisely:

```
<TCOORD:(109041,DCM,"P wave")=(POINT,"0.735")>
```

A dump of the actual DICOM attribute encoding, would look as follows:

```
(0040,a040) Value Type "TCOORD"  
(0040,a043) Concept Name Code Sequence  
(ffff,e000) Item  
(0008,0100) Code Value "109041"  
(0008,0102) Coding Scheme Designator "DCM"  
(0008,0104) Code Meaning "P wave"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0040,a130) Temporal Range Type "POINT"  
(0040,a138) Referenced Time Offsets "0.735"
```

In this example, the temporal coordinate is a single point, specified as a time offset from the beginning of the waveform or image data, in seconds.

A TCOORD content item will always have one or more children that are of either SCOORD, IMAGE or WAVEFORM value type.<sup>82</sup> It is not possible to use a TCOORD to refer to an arbitrary point or range in time outside the context of an image or waveform.<sup>83</sup> Children are included by adding a Content Sequence, as shown previously for the text value type.

When a TCOORD value type content item is used it may have a concept name, which specifies the purpose of the reference to the temporal coordinates.

The choices of temporal range type available are:

- ▶ POINT, which specifies a single point in time
- ▶ MULTIPOINT, which specifies multiple, independent points in time
- ▶ SEGMENT, which specifies a range between two points
- ▶ MULTISEGMENT, which specifies multiple ranges, each between two points
- ▶ BEGIN, which specifies the beginning of a range to beyond the end of the data
- ▶ END, which specifies the end of a range that starts before the beginning of the data

---

82. The SCOORD, IMAGE or WAVEFORM children will also always have a SELECTED FROM relationship with the TCOORD parent, as described later.

83. It would be possible to specify such a time, given the encoding mechanisms available, but it is expressly forbidden by the requirement that the content item have at least one child that is an image or waveform. This restriction may seem a bit harsh, but one can always use a content item with a DATETIME value type to refer to a point in time outside the context of an image or waveform.

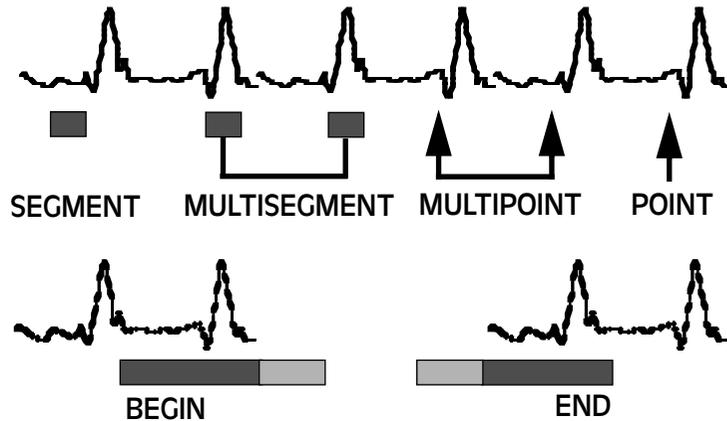


FIGURE 11. Temporal Range Types for Temporal Coordinates

The temporal data may be encoded in one of three, mutually exclusive forms:

- ▶ as one or more sample offsets, encoded as 32-bit unsigned binary integers (UL) numbered from 1 for the first sample, in Referenced Sample Positions (0040,A132)
- ▶ as one or more time offsets from the start of data acquisition, in seconds, encoded as decimal strings (DS) in Referenced Time Offsets (0040,A138)
- ▶ as one or more absolute date-times, encoded with a DT value representation in Referenced Datetime (0040,A13A)

Only a time offset or referenced date-time can be used to indicate a temporal coordinate in a multi-frame image, since there are no “samples” in images to point to. Any one of the three forms may be applied to waveforms.

For example, given

- ▶ an acquisition starting at exactly 10 am on the 3<sup>rd</sup> of February, 2000 (MET)
- ▶ a multi-frame image with a one second frame interval
- ▶ a synchronously acquired waveform with a 100 millisecond sampling rate

then a temporal point that is at the beginning of the 1st frame and first waveform sample can be indicated by:

- ▶ a TCOORD with a referenced time offset value of zero seconds
- ▶ a TCOORD with a referenced date time of “20000203100000.000+0100”
- ▶ a TCOORD with a sample position of one, for the waveform only
- ▶ an IMAGE reference to the first frame of the image only

Similarly, a temporal point that is at the beginning of the 10th frame, which corresponds to the 101st waveform sample, can be indicated by:

- ▶ a TCOORD with a referenced time offset value of 9 seconds
- ▶ a TCOORD with a referenced date time of “20000203100009.000+0100”

- ▶ a TCOORD with a sample position of 101, for the waveform only
- ▶ an IMAGE reference to the 10<sup>th</sup> frame of the image only

The next waveform sample could be indicated by:

- ▶ a TCOORD with a referenced time offset value of 9.1 seconds
- ▶ a TCOORD with a referenced date time of “20000203100009.100+0100”
- ▶ a TCOORD with a sample position of 102

Which mechanism to choose depends on a number of factors, including:

- ▶ whether the same temporal reference is to apply to both images and waveforms, in which case neither the sample offset nor the image frame number reference can be used
- ▶ whether the (start of the) waveform or image acquisition is synchronized to an accurate (real-world) clock, in which case the referenced date time may be used<sup>84</sup>
- ▶ whether harmonization with other waveform annotation standards, such as HL7, is required, in which case the time offset may be a better choice

An application that receives structured reports containing temporal coordinates is required to be able to support all three mechanisms, since the choice is at the discretion of the creator of the report.<sup>85</sup>

## Purpose of Reference

The use of Concept Name (the “name” component of the “name-value” pair that constitutes a content item) is specialized for all object reference and coordinate value types, including:

- ▶ references to image objects (“IMAGE”)
- ▶ references to waveforms (“WAVEFORM”)
- ▶ references to other composite objects, such as reports (“COMPOSITE”)
- ▶ references to DICOM Unique Identifiers (“UIDREF”)<sup>86</sup>
- ▶ spatial coordinates (“SCCOORD”)

---

84. The start of the acquisition of waveform data is reliably specified in the waveform object by the mandatory (type 1) Acquisition Datetime (0008,002A) attribute. Traditionally, it has been difficult to reliably convey the start of the acquisition of image data. Vendors have varied in their interpretation of whether to encode the start of acquisition (as opposed to pixel data creation) in Content (formerly Image) Time (0008,0033) or Acquisition Time (0008,0032). Accordingly, as of Supplement 30 (Waveform), the Acquisition Datetime (0008,002A) attribute has been added as an optional attribute to all images, with the same definition as for waveforms. The Acquisition Time Synchronized (0018,1800) attribute of the Synchronization Module indicates whether or not the specified date-time has been synchronized to an external time reference, such as a GPS (Global Positioning Satellite) or NTP (Network Time Protocol) reference.

85. In the case of the referenced date-time, the receiver should also be prepared to handle the case when the time zone of the reference is different from the time zone of the start of acquisition of the data.

- ▶ temporal coordinates (“TCOORD”)

For these value types, the concept name has the specific meaning “purpose of reference.” The exact meaning of “purpose of reference” is not defined by the standard. The intent is probably best explained by way of some examples.

In the case of UIDREFs to a study, potential purposes of reference might include:

- ▶ reference to the study that is the source of the images being reported upon
- ▶ reference to a relevant prior study

In the case of COMPOSITE object references, potential purposes of reference might include:

- ▶ reference to a prior report on a prior study
- ▶ reference to another report, part of whose content is quoted in the current document
- ▶ reference to a radiotherapy plan that is the subject of this report

In the case of IMAGE and WAVEFORM references (or SCOOD or TCOORD value types), potential purposes of reference might include generic concepts such as:

- ▶ indication of a baseline
- ▶ indication of the best illustration of a finding (that is a parent of the reference)

The notion of being able to identify an image (or frame) that is of interest from a larger set of (less interesting) images, a so-called “key image,” is a particularly popular feature of structured reporting.<sup>87</sup>

The purpose of reference for coordinates will often be much more specific, such as:

- ▶ indication of a specific anatomic feature
- ▶ indication of a specific functional or physiological feature
- ▶ indication of a specific pathological feature
- ▶ indication of a device

For example, the purpose of reference for a set of coordinates might be “femoral artery”, “P wave”, “mass”, or “stent.”

In the case of waveforms that are voice audio recordings, the purpose of reference might convey that the waveform is:

- ▶ the entire report

---

86. This is not strictly true. In Supplement 23, though all the other reference value types are described as having a concept name that means “purpose of reference,” the UIDREF value type concept name is described only as “Type of UID, e.g ‘Study Instance UID.’” A correction proposal (CP 219) suggests that this description should be change to “purpose and type of reference.”

87. So much so that the Year 2 IHE technical framework specifically calls out this feature for demonstration, referring to it as a “key image note.” See also the chapter on “Templates: Bringing order to chaos ...”.

- ▶ the whole or part of an original dictated report that has been transcribed
- ▶ a voice annotation of a specific section of a report, such as a conclusion or impression

One is not required to specify the purpose of reference. That is, one may omit the concept name for any object reference or coordinate value type. Indeed, since coordinates always have a child that is an object reference,<sup>88</sup> it usually makes more sense to specify the purpose of reference only for the coordinate content item (which is higher up in the tree), and omit it for the child object reference (which is lower down in the tree). An example is shown in Figure 12.

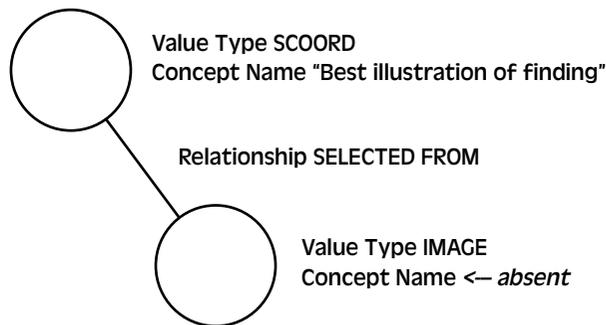


FIGURE 12. Purpose of Reference for Coordinates and Images

If one specified these the other way around, tagging the IMAGE rather than the SCORD with the purpose of reference, then the meaning of the coordinates might be ambiguous, and the purpose of reference to the image excessively specific.

### Only Simple Types

From the foregoing, it should be apparent that each content item contains what is essentially a “simple” data type. That is, each node contains a single value. One could imagine that more complex value types could be defined which contained:

- ▶ multiple values of the same type, like arrays
- ▶ multiple values of different types, like structures<sup>89</sup>
- ▶ value types for tables with rows and columns and headings

and so on.

The design philosophy of SR is to defer the encoding of all such “aggregate” data types to the document tree, and to use coded entries and relationships to build complex

88. With a *SELECTED FROM* relationship, which is defined in the chapter on “Encoding & Relationships: Building the Tree.” The use of this relationship and the purpose of reference will be discussed further in the later chapter on “Images, Waveforms, and Voice Audio.”

89. As used in programming languages like C, C++ or Java.

structures from the simple atomic types. This does mean, however, that consistency of encoding of complex data will depend on the adoption of “templates” that describe common patterns of encoding. The encoding of the tree and relationships will be discussed in the next chapter. A subsequent chapter will discuss templates. Examples of the use of the tree to encode common patterns like two-dimensional tables of information will also be described later.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ each content item consists of a name-value pair
- ▶ the name is encoded as the concept name; it is always a coded entry rather than plain text
- ▶ containers are content items without values and a value type of CONTAINER; they may be named, in which case the name is a heading
- ▶ containers do not have to be headings (they do not have to have a concept name)
- ▶ the root node is always a container, and it’s concept name is the document title
- ▶ nested containers provide a mechanism to encode multiple levels of headings and sub-headings
- ▶ the value of non-container content items may be one of a range of value types
- ▶ a value type of TEXT is used for unformatted text
- ▶ a value type of CODE is used for coded entry values, in which case both the name *and* the value of the name-value pair are coded entries
- ▶ a value type of NUM is used for numeric measurements; these measurements always have units (which are coded entries)
- ▶ a value type of PNAME is used for people’s names
- ▶ a value type of DATE is used for dates (of entire days)
- ▶ a value type of TIME is used for times (on any day)
- ▶ a value type of DATETIME is used for times on a particular date
- ▶ a value type of UIDREF is used to encode the identifier of an individual, normalized, DICOM entity, but is never used for composite objects such as reports, images and waveforms
- ▶ the TEXT, CODE, NUM, PNAME, DATE, TIME, DATETIME and UIDREF value types all require a concept name
- ▶ references to image objects are encoded with a value type of IMAGE; these may specify not only an image but a frame or range of frames, as well as a presentation state
- ▶ references to waveforms objects are encoded with a value type of WAVEFORM; these may specify not only a waveform, but a channel or channels in a multiplex group or groups

- ▶ spatial coordinates on referenced images are specified with a value type of SCOORD, which may define a point, multiple points, a closed polygon, a circle or an ellipse
- ▶ there is no mechanism to specify an arbitrary location in a patient-relative three-dimensional space, without reference to an image
- ▶ temporal coordinates on referenced waveforms or images are specified with a value type of TCOORD, which may define a point, segment, multiple points or segments, or the beginning or end of a segment, either as a sample offset, time offset or absolute date and time
- ▶ temporal coordinates cannot be used to reference an absolute date and time outside the context of an image or waveform; a DATETIME value type should be used for that purpose
- ▶ references to other composite objects, such as reports, are specified with a value type of COMPOSITE
- ▶ the COMPOSITE, IMAGE, WAVEFORM, SCOORD and TCOORD value types may have a concept name, but are not required to
- ▶ when a concept name is present for UIDREF,<sup>90</sup> COMPOSITE, IMAGE, WAVEFORM, SCOORD and TCOORD value types, it encodes the purpose of the reference
- ▶ all value types are simple, atomic types; complex types like lists, arrays, structures and tables are encoded using multiple content items in the document tree

---

*90. This is not strictly true for UIDREF as noted earlier.*

# ***Encoding & Relationships: Building the Tree***

---

A structured report would not be very interesting if it merely consisted of a flat list of content items; it wouldn't really be "structured" at all. Consequently a structured report consists of a tree of hierarchical information, such that information "higher" in the tree is in some way derived from, predicated upon, or contains, information "lower" in the tree.

Constructing a tree requires:

- ▶ a method of encoding the data structure that represents the tree
- ▶ a definition of what it means to be located somewhere in the tree, specifically what it means to be "related" to one's ancestors, siblings and descendants

The encoding is the easy part, and will be discussed first. The definition of relationships is the fun, hard and useful part. Discussion of what relationships are specified, what they mean, and how to use them, will form the bulk of this chapter.

## **Encoding a Tree**

Programmer's have a plethora of tools for encoding trees of various shapes, sizes and descriptions. The choice of a particular tool is often predicated on the application of the tree. One choice may provide for efficient traversal in one direction or another. Another choice may allow for natural sorting based on some pre-specified criteria. Yet another choice may allow for efficient insertion of updated information. And so on. In the case of the DICOM structured reporting tree, the primary application is document interchange. That is, the encoded document is an externalized (or "serialized") representation of some information. How one represents a DICOM SR document internally in an application, either literally in memory or through some API,<sup>1</sup> is up to the implementer, and not the standard.<sup>2</sup>

Accordingly, the constraints and requirements that were considered in choosing an encoding for the SR tree included the following:

- ▶ clarity and simplicity had priority over compactness
- ▶ existing DICOM mechanisms of encoding had to be used wherever possible, to avoid redesign, and to make use of existing experience and tools
- ▶ parsing into an internal representation had to be easy
- ▶ writing an internal representation out to a DICOM object had to be easy
- ▶ efficiency of access to a particular node had a low priority
- ▶ ease of updates “in place” in a serialized form had a low priority
- ▶ the structure had to be extensible to a more complex graph than a tree<sup>3</sup>

Two alternative forms of encoding were considered:

- ▶ a flat list of content items, with the root node explicitly identified, and the entire tree structure encoded by pointers to implicit or explicitly labelled content items<sup>4</sup>
- ▶ a recursively nested encoding that mirrors the logical tree structure

The recursively nested encoding was chosen. The existing DICOM sequence attribute mechanism lends itself well to recursive nesting, allowing as it does sequence of items, each of which can contain any attributes, including further sequence attributes, ad infinitum.<sup>5</sup>

It was decided that the top level content item (the root node) would have its attributes “exposed,” rather than be nested as a single item of a single sequence attribute. This was to facilitate access during a query to the Concept Name Code Sequence (used as the document title). Nested sequence item matching is not one of DICOM’s better features.

Accordingly, the root node of a report with no nested content is encoded something like this:<sup>6</sup>

```
... top level attributes from general modules ...
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
```

- 
1. *Application Programmer Interface: a specified family of data structures, function calls and objects that together provide some sort of related functionality, such as the boundary between an application and the operating system services that it requires.*
  2. *See the later chapter on “Trees, Traversal and Transformation” for further discussion of implementation issues.*
  3. *This requires an ability to identify (“label”) nodes and then reference them from elsewhere within the tree.*
  4. *This would have been similar to the form that was chosen for the DICOMDIR.*
  5. *The standard places no limits on the allowed depth of recursion, even though (before SR) objects had never used more than a few nesting levels. Toolkit vendors seemed to have implemented general rather than depth limited solutions, so it was not expected that this would be a problem.*
  6. *Remember that the root node is always a CONTAINER value type.*

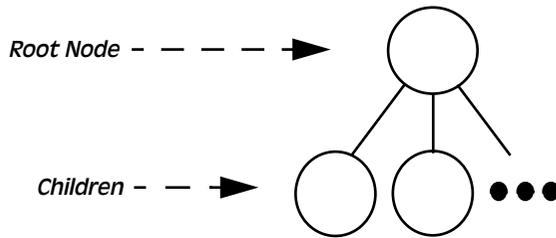


FIGURE 13. Example of a Simple SR Tree with One Level of Children

```

(0008,0100) Code Value "209076"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Chest X-ray Report"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
... more top Level attributes ...
  
```

Notice how the Concept Name Code Sequence is exposed as a top level attribute.

Some content can be added to this report to create a tree, as illustrated in Figure 13. To encode this tree, the children of the root node are added using a Content Sequence (0040,A730) containing one or more items, each of which is itself a "content item" (another "node"):

```

... top level attributes from general modules ...
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209076"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Chest X-ray Report"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
    ... first child goes here ...
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
    ... second child goes here ...
(fffe,e00d) Item Delimitation Item
    ...
(fffe,e0dd) Sequence Delimitation Item
... more top level attributes ...
  
```

Note that there must always be at least one child (item) of a content sequence ... there wouldn't be any point in encoding an empty content sequence.<sup>7</sup> If the content

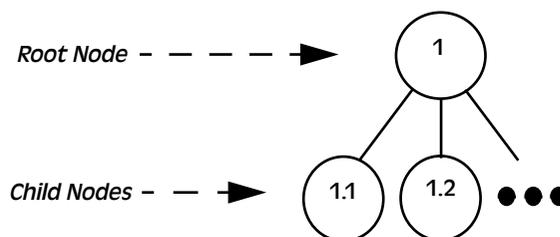


FIGURE 14. Implicit Numbering of Nodes in the SR Tree

sequence is absent then there are no children and the node is a “leaf” or “terminal node.”

The order of the items in the content sequence is deemed to be significant.<sup>8</sup> A DICOM parser is required to maintain the order of items in a sequence as they are analyzed.<sup>9</sup> The first item encountered in the encoded bit stream is defined to be the first child, and so on. Ordering is not always important from a semantic perspective.<sup>10</sup> For example, it is not important when conveying a list of descriptors. In other cases order is important, such as when conveying the headings of an outline.<sup>11</sup>

There is no explicit encoding or labelling of the order. The first child (sequence item) is implicitly numbered “1” and the second “2” and so on. This is important, because the implicit numbering of content items is used in the referencing mechanism that allows one to define a data structure more complex than a tree. This will be discussed

7. *The Content Sequence attribute is defined with a requirement of type 1C with a condition that “one or more items may be included in the sequence.” This is the pattern used in the standard to forbid a zero length or empty sequence.*
8. *In fact, the order of items is significant in any DICOM sequence attribute. This was a controversial issue. It had always been intended that this be the case, but nothing in the standard explicitly specified it (apart from the name “sequence” rather than say “set” - ASN.1 makes this distinction). The Nuclear Medicine object’s multi-frame descriptors depend on implicit ordering of sequence items, but no-one had given the issue much thought. It would be possible (indeed perhaps easier) to build a toolkit that ignored or reversed the order of items (using a stack rather than a FIFO). In any case, it was decided during the SR and waveform discussions to bite the bullet, and once and for all explicitly specify that sequence items are ordered.*
9. *The parser can represent the items any way it wants, as long as it retains the ability to reconstruct the order.*
10. *There is no mechanism, by the way, to specify that order “doesn’t matter” for a particular set of siblings.*
11. *It can be argued that ordering of content items is a presentation related issue, and outside the scope of SR. It is really the job of the rendering application to apply some sort of style rule that says, for example, that conclusions should be presented after findings (or vice versa), and so on. However, in the case of sequential blobs of plain text, each equivalent to the conventional notion of a “paragraph” perhaps, it would be difficult to express this with a style sheet alone. For better or for worse, implementers are going to depend on sequence item ordering to get the presentation they expect. See also the discussion of the “Continuity of Content” attribute, which is related to this.*

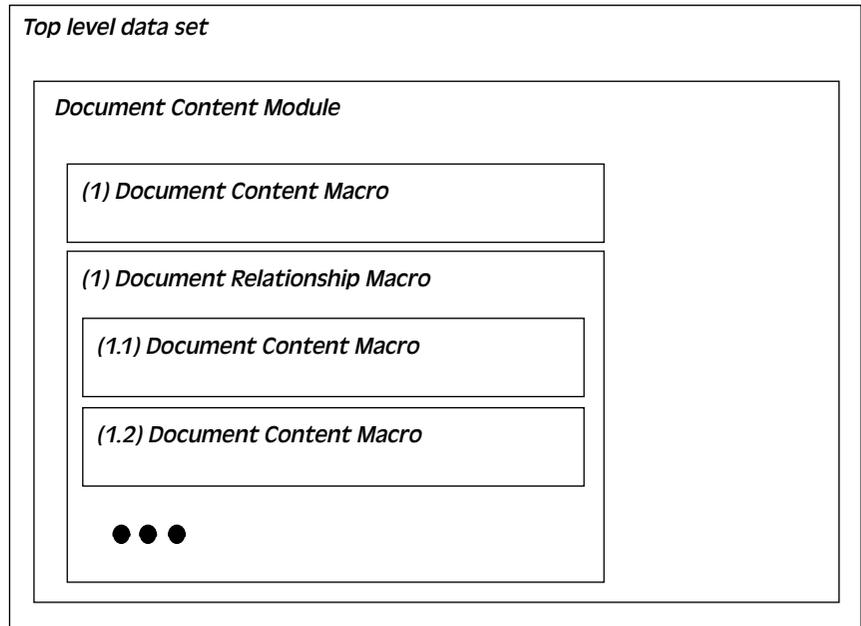


FIGURE 15. Example of Nesting of Document Macros

in more detail when “by-reference” relationships are described. For now it is sufficient to say that the root node is always “1”, the first child of the root is “1.1”, the second child of the root “1.2”, etc. This is illustrated in Figure 14.

This nested sequence encoding of the SR tree is not immediately obvious when looking at the standard. The SR IODs are defined to contain a SR Document Content Module, which consists of two “macros.”<sup>12</sup>

- ▶ the Document Relationship Macro
- ▶ the Document Content Macro

The Document Content Macro defines the name and value that comprise the name-value pair that is the “payload” of the content item.

The Document Relationship Macro, in addition to describing some “administrative” attributes like Observation Datetime, is where nested (or referenced) content is defined. It is in this macro that the Content Sequence attribute occurs. Within the Content Sequence attribute, the Document Content Macro (and the Document Rela-

12. Macros have recently been added to the standard as a documentation convention. They were originally conceived in order to simplify writing coded sequence entries. Rather than having to write the same triplet of attributes every time an attribute was defined as a code sequence, a macro could be invoked. The notion of recursively invoking macros is a logical extension. It is a bit more tricky to implement in a validating parser that tries to check compliance of an object with the standard. None of this makes any difference to how an object is actually encoded. There are no macros in the bit stream. They are merely a documentation tool.

tionship Macro if there are further nested children) are recursively invoked. Figure 15 illustrates a simple example of a root node with two or more children, from the perspective of how the macros are recursively invoked.

The Document Relationship Macro also specifies that the items of the Content Sequence attribute are required to contain a Relationship Type attribute, which will now be discussed, in exhaustive (or exhausting) detail.

## Relationships

Up to this point, the encoding of the SR tree has been presented from a purely syntactic point of view. What has not been discussed is what it actually “means” to be a parent, a child or a sibling. That is, what semantics are implied or expressed by the data structure.

It is in this respect that DICOM SR diverges from other structured document approaches. *There are no implicit semantics inherent in the tree encoding whatsoever.* In other approaches, such as XML, the notion of “containment” is implied and inherent in the nested encoding.<sup>13</sup> DICOM SR does not imply this: containment must be explicitly encoded.

The explicit attachment of semantics to the tree is achieved through the definition of “relationships” between nodes (content items).

A relationship between two nodes requires designation of:

- ▶ the “source” node (content item) of the relationship
- ▶ the “target” node (content item) of the relationship
- ▶ the type of relationship

That is, one specifies that “Node S” “has relationship R” with “Node T.” As a consequence of the DICOM encoding of the SR tree, the source node of a relationship is always the parent node. For relationships “by value,” the target node is always the child of that parent.<sup>14</sup> The type of relationship is encoded with the child, so that a parent may have different relationship types with each of its children.

The type of relationship is encoded as described by the Document Relationship Macro in the attribute Relationship Type (0040,A010). For example, if a node contains two children, one with an INFERRED FROM relationship type and the other with a HAS PROPERTIES relationship type, they would be encoded as follows:

13. For example, the XML fragment “<name><first>david</first><last>clunie</last></name>” defines not only the values of the “first” and “last” name elements. The fragment also defines that they are semantically “contained” within the “name” element. That is, the first name is “part of” the name.

14. The other type of relationship, relationships by reference, will be discussed later. Relationships by value are the only kind of relationships possible when encoding a structure that is strictly a tree, rather than a more complex graph. They are the only kind of relationships that are allowed in the Basic and Enhanced SR SOP classes.

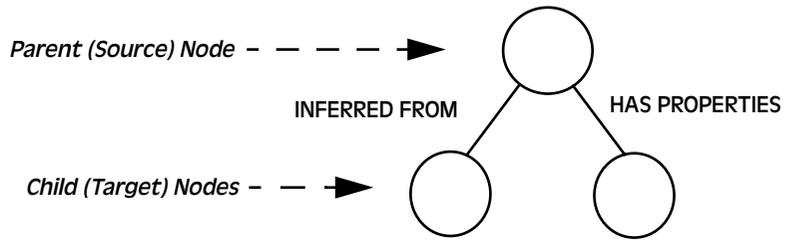


FIGURE 16. Diagram Style to Illustrate SR Trees - Vertical

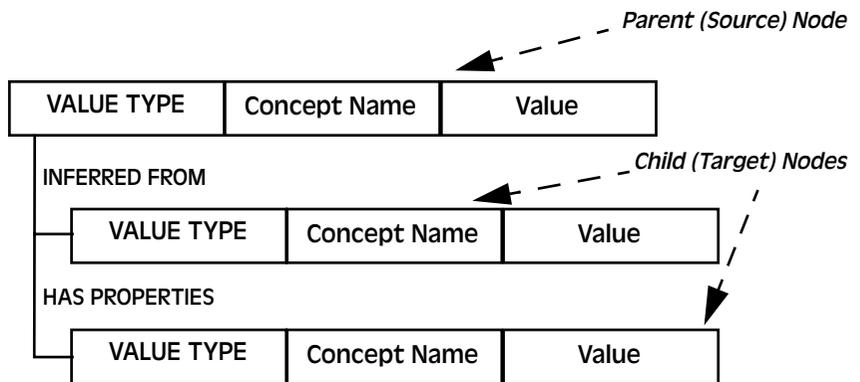


FIGURE 17. Diagram Style to Illustrate SR Trees - Horizontal

```

(0040,a040) Value Type “...”
    ... rest of parent node ...
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type “INFERRED FROM”
    ... content of first child goes here ...
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a010) Relationship Type “HAS PROPERTIES”
    ... content of second child goes here ...
(fffe,e00d) Item Delimitation Item
    ...
(fffe,e0dd) Sequence Delimitation Item
    ... rest of parent node ...
    
```

Two styles of diagram will be used to illustrate relationships. The first is drawn to look like a tree, with circles for nodes, and arcs (that are actually straight lines) between them to illustrate relationships. The arcs are labelled with the relationship type. The form is shown in Figure 16. This style of drawing, though it looks like the tree that it represents, gets a little unwieldy when more than a few nodes are present.

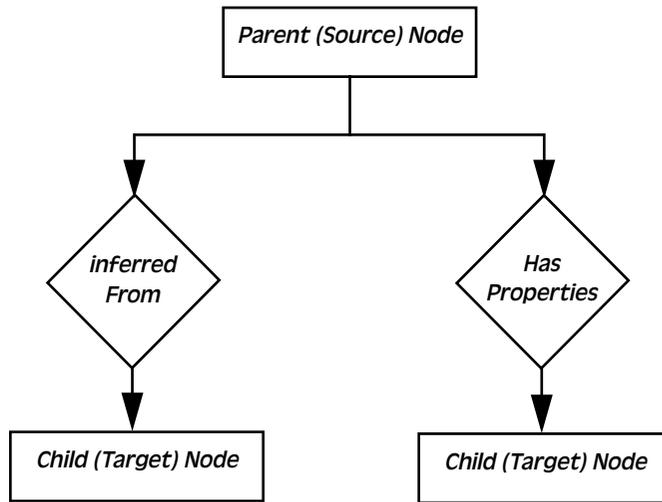


FIGURE 18. Diagram Style to Illustrate SR Trees - E-R Form

Accordingly, a “horizontal” representation of the tree will be used to illustrate examples that are more than trivial as shown in Figure 17.<sup>15</sup>

The style that is used in the standard to represent structured report trees uses entity-relationship (E-R) diagrams. This style is bulky for more complex examples, using as it does rectangles for the content items and diamonds for the relationships between them. An example of the form is shown in Figure 18.

## Relationship Types

Just as there is an enumerated set of value types for specifying the encoding of content item values, so there is an enumerated set of “relationship types.”<sup>16</sup> These are encoded in the attribute Relationship Type (0040,A010) which is present in every content item that is a child of another content item.<sup>17</sup> The following relationship types are currently defined and used in SR SOP classes:<sup>18</sup>

- ▶ contains (“CONTAINS”)

15. Charles Parisot was the first to suggest using this style for the purpose of slide and video projector presentations. It makes better use of the space available in a landscape-oriented page.

16. Strictly speaking, both value types and relationship types are declared in the document content and relationship macros to be “defined terms.” In reality however, both are constrained to enumerated values by the SOP classes that are currently standardized. The modules form the building blocks of current and future IODs for SOP classes. They are deliberately generic in case the need arises to add new value types or relationship types. The use in the modules of the phrase “defined term” rather than “enumerated value” is in essence a warning to implementers to allow for the possibility of new terms in the future.

17. In other words, all content items except the root require a relationship type attribute.

- ▶ has properties (“HAS PROPERTIES”)
- ▶ inferred from (“INFERRED FROM”)
- ▶ selected from (“SELECTED FROM”)
- ▶ has observation context (“HAS OBS CONTEXT”)
- ▶ has acquisition context (“HAS ACQ CONTEXT”)
- ▶ has concept modifier (“HAS CONCEPT MOD”)

This is not a very long list. During the development of SR, many other possibilities for encoding relationships were proposed.

At one point it was suggested that relationship types should be coded entries, to allow for a plethora of highly specific relationships (like “has size,” “has margin,” “has color” and so on). It was realized, however, that most of those kinds of relationship would be redundant with the concept name of the target of the relationship. In essence what has been adopted is a scheme whereby by a statement like:

name-value “has named property” value

has been factored out as:

name-value “has property” name-value

This approach allows for a smaller set of relationships, and reduces redundancy with concept names. Otherwise, an entire set of coded entries would need to have been developed for all the property relationships that would have shadowed the corresponding concept names.

The three primary relationship types for encoding content are “contains,” “inferred from” and “has properties”:

- ▶ the “contains” relationship specifies that a child and all its descendants are contained within or are part of its parent; containment is often used as a means of conveying headings and sub-headings within a document;<sup>19</sup> the rationale being that, for example, the “introduction,” “body” and “conclusion” of a document are “contained within” the greater whole that is the document itself
- ▶ the “has properties” relationship specifies that the parent (source) has some characteristics, like size or shape or whatever; the nature and value of the property is conveyed by the child (target), and, if it is a complex property, all of the child’s descendants; for example, a mass may have properties that include size, shape, margin, etc.
- ▶ the “inferred from” relationship specifies that the parent (source) is a conclusion, deduction or inference which has been made from the information that is described by the child (target) and its descendants; for example, that a finding on a

---

18. The Code String value representation is limited to 16 characters and upper case, which is why some of the relationship types (shown in parentheses) are abbreviated and may seem a little hokey.

19. In earlier DICOM SR literature, “headings” were referred to as “categories.”

mammogram is potentially malignant may be inferred from the presence of a certain type of clustered micro-calcifications

Which relationship type to choose to encode content depends a lot on the style and application of the document in question. On the one hand, when encoding the structure of a plain text document that has clearly discernible headings, “contains” relationships with “container” value types will likely be used extensively. On the other hand, when encoding the logic behind the conclusions of a computer-assisted-diagnosis device, the “inferred from” relationship would be appropriate. When properties of an object can be clearly discerned, whether it be in the course of natural language parsing of dictated plain text, as a consequence of a user’s selection from a “pick-list,” or as a result of an automated or semi-automated measurement procedure, then use of the “has properties” relationship would be appropriate.

Each of these relationship types will be described in greater detail later in this chapter.

The other relationship types are a little different, and have highly specific patterns of use, as well as semantics defined by the standard that go beyond merely defining content.

Once the concept of relationships was introduced, it became apparent that they provide a convenient way of factoring out and describing patterns which might otherwise have been “hard-wired” with a specific encoding. For example, when it was decided to define spatial coordinates of images, as well as references to images without coordinates, two approaches could have been taken. One approach would have been to define two separate value types, spatial coordinates and image references, with the spatial coordinate value type including all the attributes of the image references within it. What was decided instead was:

- ▶ to “reuse” the image value type
- ▶ to specify a spatial coordinate value type without any built-in image reference
- ▶ to use the “selected from” relationship to join the two

This choice made it possible to reuse value types in more creative ways, such as to refer to the same temporal coordinates selected from both waveforms *and* spatial coordinates selected from images, etc. A basic discussion of the use of the “selected from” relationship will be included in this chapter, and further detail will be provided in the chapters on Images, Waveforms, and Voice Audio.

Another design decision was to allow concept names to be “modified” by using the relationship mechanism, rather than to hard-wire concept modifier attributes into every content item. This tree-based approach allows for more complex modifiers and facilitates construction of both simple and complex post-coordinated concepts.<sup>20</sup>

---

20. The need for this became apparent when people started to talk about matching elements in lists of modifiers for different concepts in different content items, based on their implicit order. It became clear that the SR tree could be used for this. The waveform supplement takes the opposite approach for annotations and provides an interesting contrast (or annoying inconsistency depending on how one looks at these things). The Waveform Annotation module uses a Modifier Code Sequence attribute which is never used in SR.

The design principle that grew out of this approach is to “use the SR tree” whenever possible, and to avoid introducing specific encoding mechanisms to define complex patterns.

The “has concept modifier” relationship may be used to encode equivalent meanings in the same or different languages, as well as to construct simple or complex modifiers that qualify, clarify and/or elucidate the meaning of a specified concept.<sup>21</sup>

The “has observation context” relationship is significantly different from other relationship types. There are explicit semantics defined in the standard for “inheritance” of attributes of observation context.<sup>22</sup> This is a sufficiently complex topic that warrants its own chapter on “Context: Who did what to whom, when...”. Acquisition context, to which inheritance does not apply, is discussed in the chapter on “Images, Waveforms, and Voice Audio.”

## **Contains Relationship Type**

The “contains” relationship type has some specific uses, though it is often considered to be a generic “catch-all” relationship, the relationship one has when one is not having a relationship.<sup>23</sup> As has already been mentioned, the concept of “containment,” that is of being a sub-part of a greater whole, is common to tree-structured document encoding.

The idea that a sub-section of a text-oriented document is contained within or is part of a section at a higher level is a familiar theme in documents of any kind. From this real-world paradigm comes the notion in SR that the concept name of a container value type may be a “heading” for all of the child content items that are “contained” within the “container.” In the chapter on “Content Items: Concept Names and Values,” the container value type and its use to encode headings were discussed. Here, further aspects of containers that are relevant to the “contains” relationship type will be described.

It makes sense that the source value type of a “contains” relationship always be a “container.” In fact, this is a requirement of the IODs for each of the SOP classes, even though the generic framework (defined by the document content and relationship macros) doesn’t specify such a constraint. Specifically:

- ▶ only containers can be the source of a contains relationship
- ▶ containers may have children with relationships other than contains (“has observation context,” and so on)
- ▶ the target of a contains relationship may be, but does not have to be, a container

---

21. Or perhaps even change the meaning entirely, as with a modifier used to indicate negation, for example.

22. One may well ask why observation context is singled out to be graced with an inheritance mechanism, and yet other relationship concepts are not. There is no straightforward answer.

23. A “Clayton’s” relationship, according to Australian non-alcoholic beverage advertising on television during the last century.

This last point requires some elaboration. To be consistent with the paradigm that container concept names represent headings, and that contained containers are lower level sub-headings, further restrictions are required. Yet one may want to contain both actual content (value types like “text” or “code” or “numeric”) and lower level headings within the same higher level heading.<sup>24</sup> However, once one has “stopped using headings,” so as to speak, and descended into the depths of actual content, one may wish to restrict the further use of both the container value type and the contains relationship. Indeed, the IODs for the Basic and Enhanced SR SOP classes do restrict such uses, by forbidding containers as the target value type in any situation in which the source value type is not a container.

Consider a simple case in which, as always, the document root is a container. There is one level of children that are headings, each of which contains textual content:

```
<CONTAINER:(,,“Chest X-Ray Report”)>
  <contains CONTAINER:(,,“Procedures”)>
    <contains TEXT:(,,“Procedure”)=”PA and Lateral”>
  <contains CONTAINER:(,,“Findings”)>
    <contains TEXT:(,,“Finding”)=”Multiple masses”>
  <contains CONTAINER:(,,“Conclusions”)>
    <contains TEXT:(,,“Conclusion”)=”Metastases”>
```

This is a fairly common pattern when DICOM SR is used to encode a plain text report with headings. There is no reason, however, why containers should not contain a mixture of content and other containers, why the nesting of containers used as sub-headings be consistent, nor why there need be any sub-headings, as illustrated in the following example:

```
<CONTAINER:(,,“Chest X-Ray Report”)>
  <contains TEXT:(,,“Procedure”)=”PA and Lateral”>
  <contains CONTAINER:(,,“Findings”)>
    <contains TEXT:(,,“Finding”)=”Multiple masses”>
  <contains CONTAINER:(,,“Conclusions”)>
    <contains CONTAINER:(,,“Impressions”)>
      <contains TEXT:(,,“Impression”)=”Metastases”>
    <contains CONTAINER:(,,“Recommendations”)>
      <contains TEXT:(,,“Recommendation”)=”Biopsy”>
  <contains CONTAINER:>
    <contains CODE:(,,“Diagnosis”)=(197.0,I9C,
      “Secondary malignant neoplasm of lung”>
```

Notice in this rather contrived example that the root node contains content in the form of text, not just containers, that the conclusions container contains further nested containers though the findings container does not, and that the ICD9 diagnosis code is contained in a container with no concept name (no heading).

---

24. This situation is analogous to technical documents in which a “section” may have an introductory paragraph or two “of its own,” before the first “sub-section” actually starts.

For the sake of completeness, how this example would be encoded at the attribute level will be shown.<sup>25</sup> Remember from the preceding sections that the relationship type is encoded within each child content item, and that recursive nesting of content sequence attributes is used to encode child content items. Also notice how the container that is the last child of the root node has no concept name code sequence, which is allowed for containers;

```
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209076"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Chest X-ray Report"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209007"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Procedure"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "PA and Lateral"
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209002"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Findings"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
```

---

25. Don't be too intimidated by the length of this example. The low-level encoding of every example won't be reproduced here, but it is important to see one or two in order to get a sense of how things fit together.

```

(0008,0100) Code Value "209001"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Finding"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "Multiple masses"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209006"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Conclusions"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209013"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Impressions"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209014"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Impression"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "Metastases"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence

```

```

(fffe,e000) Item
(0008,0100) Code Value "209015"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Recommendations"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "TEXT"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209016"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Recommendation"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a160) Text Value "Biopsy"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a050) Continuity of Content "SEPARATE"
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CODE"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209017"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Diagnosis"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "197.0"
(0008,0102) Coding Scheme Designator "I9C"
(0008,0104) Code Meaning "Secondary malignant neoplasm
of lung"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

```

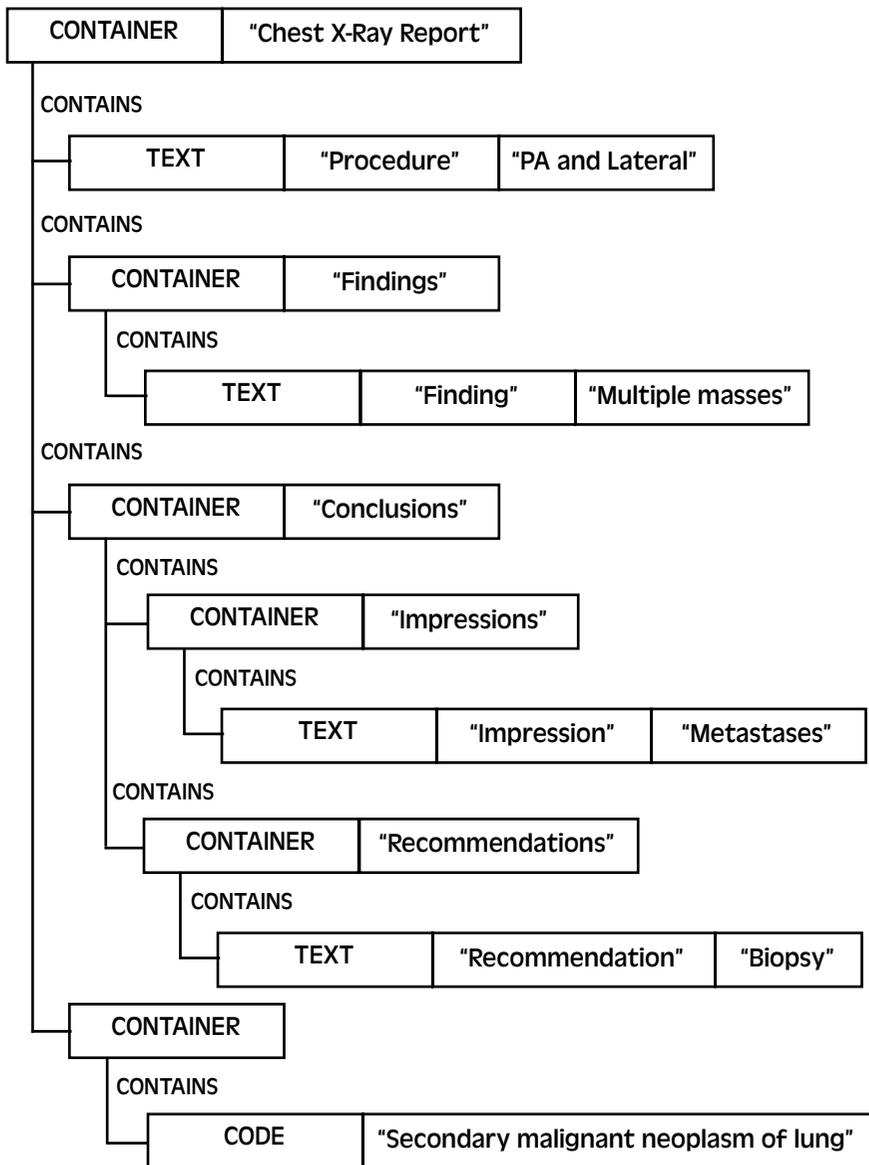


FIGURE 19. Graphical Example of a Text SR with Headings

This is pretty horrifying to deal with, so the same thing graphically as shown in Figure 19.

It is important to remember that containers and contains relationships are not just for headings. Indeed the concept of “containment” has been stretched a little to even permit this usage. An instance of “real” containment might be the specification of the components of a more complex attribute than can be described by one of the basic

SR value types. For example, to specify a US street address, one might use the following approach:

```
<CONTAINER:(,, "Address")>
  <contains TEXT:(,, "Number")="111">
  <contains TEXT:(,, "Street")="East 14th Street">
  <contains TEXT:(,, "Town")="New York">
  <contains TEXT:(,, "Zipcode")="10003">
  <contains TEXT:(,, "State")="New York">
  <contains TEXT:(,, "Country")="USA">
```

Unfortunately, this approach can really only be used in those SOP classes that allow containers to be children of value types other than containers. At the present time, this is only allowed in the Comprehensive SOP class.<sup>26</sup>

As an aside, using this pattern of containers is similar to the XML approach of using nested elements. Indeed, in XML the nesting of elements is actually defined to implicitly mean “containment.” There is no explicit relationship mechanism in XML. The following example is an XML equivalent of the same street address pattern:

```
<Address>
  <Number>111</Number>
  <Street>East 14th Street</Street>
  <Town>New York</Town>
  <Zipcode>10003</Zipcode>
  <State>New York</State>
  <Country>USA</Country>
</Address>
```

## Has Properties Relationship Type

The “has properties” relationship is the second most commonly used relationship after the “contains” relationship. The has properties relationship has specific semantics attached to its use: it defines that the parent (source node) has some particular characteristic, such as size or shape. The nature and value of the property is conveyed by the child (target node) of the relationship. If the property is complex rather than atomic, the child node may itself be the head of a tree that describes the complex property.

For example, a mass may have simple atomic properties like diameter, shape, and margin, which could be specified as follows:

```
<CODE:(,, "Finding")=(,, "Mass")>
  <has properties NUM:(,, "Diameter")="1.3" (,, "cm")>
  <has properties CODE:(,, "Shape")=(,, "Round")>
  <has properties CODE:(,, "Margin")=(,, "Well-defined")>
```

26. One could argue that the sub-components of an address could be “properties,” and make them children of a non-container value type using a “has properties” relationship, and make the parent a text value type with nothing very much in the value field, but this is a very contrived solution.

The power of the “has properties” relationship lies in its ability to use standard sets of codes to identify properties that are common to different entities. For example, the same coded concepts may be used to describe the size of an object, regardless of whether it is a mass or a calcification or a normal anatomical part. Consistent use of standard coded properties can greatly enhance a system’s ability to search and analyze the content of structured documents.

This is not to say that the “has properties” relationship is restricted to use with coded value types, whether as source or target content items. For example, it is perfectly valid to say:

```
<TEXT:(,,“Finding”)=(,,“The nasty looking mass”)>
  <has properties TEXT:(,,“Location”)=”right upper lobe”>
  <has properties TEXT:(,,“Size”)=”pretty big”>
```

which is (marginally) more useful than just a plain text blob:

```
<TEXT:(,,“Finding”)=(,,“The nasty looking mass located in
  the right upper lobe is pretty big”)>
```

Indeed, even the relatively primitive Basic Text SR SOP class permits the use of the “has properties” relationship, with a source value type of text and target value types that include text and codes, as well as dates, times and names, amongst others.

It is also possible to specify references to objects (including images and waveforms) as the targets of “has properties” relationships, which leads to the following patterns:

```
<CODE:(,,“Finding”)=(,,“Mass”)>
  <has properties IMAGE:(,,“Illustrated by”)=...>
```

or:

```
<TEXT:(,,“Finding”)=”nasty looking mass”>
  <has properties IMAGE:(,,“Seen on”)=...>
```

In this example, while the image is not exactly a “property” of the finding per se, there really is no other way to express this concept using the relationship types available. One could argue that a “contains” or an “inferred from” relationship might be more appropriate under some circumstances. However the “contains” relationship poses problems when one is deep into the tree in the simpler SOP classes. The “inferred from” relationship is only appropriate when an inference has been made from the appearance of the image, which isn’t always the case.

In this situation one depends on a good repertoire of coded “purposes of reference” to clarify the use of the “has properties” relationship.<sup>27</sup>

Similar patterns are applicable to spatial and temporal coordinates (in those SOP classes that permit them):

---

27. The “purpose of reference” is optionally encoded in the concept name of the image reference, as it is also for waveform and composite object references, as well as spatial and temporal coordinates.

```
<CODE:(,,“Finding”)=(,,“Mass”)>
  <has properties SCOOD:(,,“Epicenter”)=(POINT,1,1)>
  <selected from IMAGE:=..>
```

In this example, one can see that it is clearly appropriate to specify the epicenter of the mass as a “property.” This is usually the case, since spatial coordinates allow a much greater degree of precision in describing or localizing a finding, compared to referring to an entire image.

A clarification is probably in order with respect to using “contains” as opposed to “has properties” relationships. An example from a proposal for encoding quantitative coronary analysis results may help to illustrate this. For each individual finding one could say something like:

```
<CODE:(,,“Finding”)=(,,“Coronary Stenosis”)>
  <has properties NUM:(,,“Percent Stenosis”)="70" ("%")>
  ... other properties of this stenosis ...
```

or alternatively:

```
<CONTAINER:(,,“Coronary Analysis Result”)>
  <contains NUM:(,,“Percent Stenosis”)="70" ("%")>
  ... other features of this result ...
```

In other words, one can either describe properties of a finding, or one can list results in a container. Either is a perfectly acceptable pattern, and which to choose is largely a question of style. Notice though, that containers themselves do not have children with “has properties” relationships. In this example, it would not be appropriate to have a container with a concept name of “coronary stenosis” which had properties like “percent stenosis.”

## Inferred From Relationship Type

The “inferred from” relationship specifies that the parent (source node) is a conclusion, deduction or inference<sup>28</sup> that has been made from the information that is described in the child (target node) and its descendants.

An “inferred from” relationship can be used to identify the source of measurements:<sup>29</sup>

```
<NUM:(,,“Diameter”)="1.3" (,,“cm”)>
```

28. What precisely is the distinction between an “inference” as opposed to a “deduction” or “conclusion”? Probably there is none in this context. The standard defines the inferred from relationship as meaning that the “Source Content Item conveys a measurement or other inference made from the Target Content Items; denotes the supporting evidence for a measurement or judgment.” The American Heritage Dictionary gives definitions of inference as “the act or process of deriving logical conclusions from premises known or assumed to be true” and “something inferred.” Webster’s gives similar definitions and goes on to say that “inference” is synonymous with both “conclusion” and “deduction.” However, both “conclusion” and “deduction” also have other meanings.

29. An “inferred from” relationship type is always used for specifying the coordinates from which a measurement was derived, rather than a “has properties” relationship.

```
<inferred from SCOORD:(,,,"Endpoints")=
(MULTIPOINT,1.5,10.5,10.5,30)>
<selected from IMAGE:=..>
```

For example, the measurement might represent a property of finding:

```
<CODE:(,,,"Finding")=(,,,"Mass")>
<has properties NUM:(,,,"Diameter")="1.3" (,,,"cm")>
<inferred from SCOORD:(,,,"Endpoints")=
(MULTIPOINT,1.5,10.5,10.5,30)>
<selected from IMAGE:=..>
```

Another use of the “inferred from” relationship is to encode that a particular finding or conclusion is a direct consequence of other findings or conclusions. For example, that a mass is malignant may be inferred from the presence of an irregular margin:

```
<CODE:(,,,"Finding")=(,,,"Malignant mass")>
<inferred from CODE:(,,,"Margin")=(,,,"Irregular")>
```

Alternatively, the same statement might be expressed as:

```
<CODE:(,,,"Finding")=(,,,"Malignant mass")>
<inferred from CODE:(,,,"Finding")=(,,,"Mass")>
<has properties CODE:(,,,"Margin")=(,,,"Irregular")>
```

A more complex example is the case of a mass on a mammogram that is potentially malignant as a consequence of its association with pleomorphic clustered micro-calcifications:

```
<CODE:(,,,"Finding")=(,,,"Malignant mass")>
<inferred from CODE:(,,,"Finding")=(,,,"Mass")>
<has properties CODE:(,,,"Margin")=(,,,"Irregular")>
<inferred from CODE:(,,,"Finding")=(,,,"Calcifications")>
<has properties CODE:(,,,"Shape")=(,,,"Pleomorphic")>
<has properties CODE:(,,,"Distribution")=
(,,,"Clustered")>
```

One disturbing feature of these simple examples is that it is a little awkward to express precisely what property is the reason for the inference. For example, is malignancy inferred from the mass itself or the property of the mass that is the irregular margin? Furthermore, what about all the other characteristics of the mass, some of which may be relevant to the inference, and some of which may not. For example, a more complete description of a mass might include multiple properties:

```
<CODE:(,,,"Finding")=(,,,"Mass")>
<has properties CODE:(,,,"Shape")=(,,,"Round")>
<has properties CODE:(,,,"Location")=(,,,"Central")>
<has properties NUM:(,,,"Size")="1.5" (,,,"cm")>
<has properties CODE:(,,,"Margin")=(,,,"Irregular")>
```

of which only the irregular shape is a cause for concern.

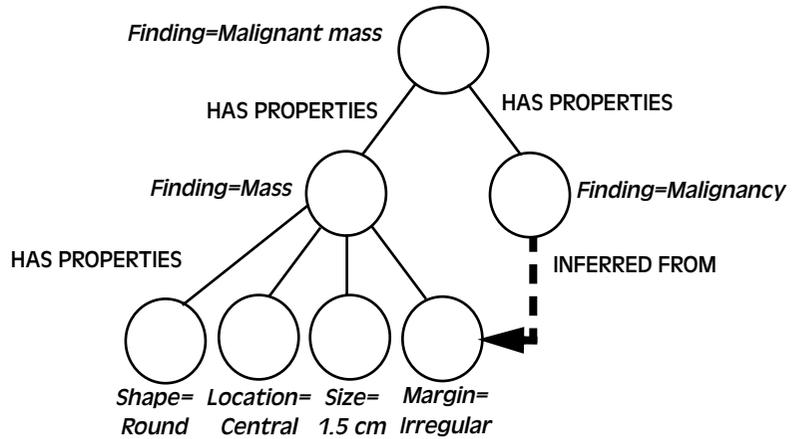


FIGURE 20. Sharing Has Properties and Inferred From By-Reference

The Basic and Enhanced SOP classes, support only by-value relationships, so there is no alternative but to replicate information to describe the mass in detail and to make inferences from a subset of properties. For example, one might encode:

```

<CODE:(,, "Finding")=(,, "Malignant mass")>
  <has properties CODE:(,, "Finding")=(,, "Mass")>
    <has properties CODE:(,, "Shape")=(,, "Round")>
    <has properties CODE:(,, "Location")=(,, "Central")>
    <has properties NUM:(,, "Size")="1.5" (,, "cm")>
    <has properties CODE:(,, "Margin")=(,, "Irregular")>
    <inferred from CODE:(,, "Margin")=(,, "Irregular")>
  
```

Using SOP classes that also support by-reference relationships<sup>30</sup> one can construct a more complex structure than a simple tree, and share a sub-group of findings between a complete description and a specific inference. For example, one can encode:

```

<CODE:(,, "Finding")=(,, "Malignant mass")>
  <has properties CODE:(,, "Finding")=(,, "Mass")>
    <has properties CODE:(,, "Shape")=(,, "Round")>
    <has properties CODE:(,, "Location")=(,, "Central")>
    <has properties NUM:(,, "Size")="1.5" (,, "cm")>
  1.1.4 <has properties CODE:(,, "Margin")=(,, "Irregular")>
    <has properties CODE:(,, "Finding")=(,, "Malignant")>
    <inferred from 1.1.4>
  
```

This example is illustrated graphically in Figure 20.

30. Amongst the general purpose SR SOP classes currently defined in the standard, only the Comprehensive SR SOP class supports by-reference relationships.

The by-reference relationship mechanism, how to describe structures that are more complex than trees, and how to label and reference individual content items will be described next.

However, the “inferred from” relationship type is very useful even when used only with by-value relationships. Typically using “inferred from” relationships by-value results in some repetition or redundancy, as in the previous example. Careful construction of the tree and judicious use of containers can make it clear and unambiguous that the same feature is being described.

Conversely, by-reference relationships may be used for other purposes than “inferred from” relationships. The bottom line is that “by-reference” and “inferred from” are *not synonymous*. The two terms should not be used interchangeably.<sup>31</sup>

## **By-Reference Relationships**

So far, this chapter has dwelt on describing information as a tree. A hierarchical organization is sufficient for many purposes. However, a strict tree often leads to repetition, redundancy or ambiguity when the same piece of information needs to be used in multiple places.

For example, the same image may be referenced from multiple content items, as the source of coordinates for two different findings. Yet it may be desirable to include a considerable amount of acquisition context about each image, such as modality, projection, laterality and so on. In a tree based model, the same context information might need to be repeated at each location where the image is referenced.

It is also sometimes desirable to be able to encode multiple “views” of the same information. For example a single document may need to contain views that describe:

- ▶ the mechanism or sequence by which information was acquired
- ▶ an outline of headings containing findings suitable for (human) review
- ▶ how inferences were made from findings which lead to conclusions

Each of these different views represents a different structural organization of the same atomic components.

To reduce redundancy and support multiple views, the SR framework allows for “pointers” or “references” to content items that are not direct descendants. Note that there is always still a “primary” tree of content, encoded as recursively nested sequence items. The difference is the additional ability to specify relationships “by-reference” instead of “by-value.”

---

31. This is emphasized because almost all the examples of the “by-reference” relationship mechanism discussed during the design of SR were examples of applications of the “inferred from” relationship type. As a consequence, many people tend to discuss the two as if they were synonymous, which they are not. A counter-example is found in Supplement 50 Mammography CAD, in which by-reference “selected from” relationships are used extensively, as are by-value “inferred from” relationships.

“By reference” relationships are actually encoded by creating a nested sequence item much like a “child.” The item contains only a relationship and a reference to a location, but no concept name, value or descendants. It is therefore not a “content item” or “child” per se, only a pointer to another content item.

Recall that “by-value” relationships are encoded by defining a source (parent) content item, a target (child) content item, and a relationship specified in the content item of the child:

```

        ... rest of parent node ...
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type “...”
        ... content of first child goes here ...
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
        ... rest of parent node ...

```

“By-reference” relationships also need to be able to specify a Relationship Type with the target, but that relationship may be different from the one that is encoded in the content item of the target itself. That is, in the place where the target content item is actually instantiated “by value” (i.e. is the child of some other parent node) it will have a relationship type with its parent encoded within its content item. That relationship may be different from the by-reference relationships that point to the target node.

Accordingly, a “by-reference” relationship is encoded as an item of the source content sequence, just like a “by-value” relationship, with an explicit relationship type, except that instead of the item containing actual content, it contains only a reference. The pattern is as follows:

```

        ... rest of source node ...
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type “...”
(0040,db73) Referenced Content Item Identifier “...”
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
        ... rest of source node ...

```

Note that there is no “content” or “value” in the item, only the relationship type and the pointer to the target node which lives elsewhere in the tree. The use of the sequence item coding is purely an encoding convenience; there is no “content item” per se in this case.

For example, consider a simplified version of the example from the preceding section:

```

<CODE:(209001,99PMP,“Finding”)=(209227,99PMP,“Mass”)>
1.1.1 <has properties CODE:(209237,99PMP,“Margin”)=
        (209238,99PMP,“Irregular”)>
<CODE:(209001,99PMP,“Finding”)=(209226,99PMP,“Malignant”)>
    <inferred from 1.1.1>

```

In this example, the node describing the margin is on the one hand descendant of the finding that is the mass, and is the “by-value” target of a “has properties” relationship. On the other hand, it is also the target of a “by-reference” “inferred from” relationship, which has a source that is the finding of malignancy. This would be encoded as follows:

```

(fffe,e000) Item
(0040,a040) Value Type "CODE"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209001"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Finding"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209227"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Mass"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a730) Content Sequence
(fffe,e000) Item <--- 1.1.1
(0040,a010) Relationship Type "HAS PROPERTIES"
(0040,a040) Value Type "CODE"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209237"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Margin"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209238"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Irregular"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e000) Item
(0040,a040) Value Type "CODE"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209001"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Finding"
(fffe,e00d) Item Delimitation Item

```

```

(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209226"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Malignant"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "INFERRED FROM"
(0040,db73) Referenced Content Item Identifier "1.1.1"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item

```

It is important to remember that “contains” relationships may not be specified by reference, only by value. In other words, content cannot be “contained” in two different sections of the same tree. A side effect of this constraint means that concept names for containers used as an outline of headings are strictly hierarchical.<sup>32</sup>

## Referenced Content Item Identifiers

In the discussion of “by-reference” relationships in the previous section, the targets of the relationships were identified by the attribute Referenced Content Item Identifier (0040,DB73). The value of this attribute was shown in the examples using numeric components separated by periods, e.g. “1.1.4”. In actuality, this attribute is encoded as multiple unsigned long values rather than a string.<sup>33</sup> Careful readers will have noticed that the target so referenced does not have an explicit label or identifier. In DICOM SR, since content items in content sequences are ordered, the path from the root node to a particular content item is always known. An implicit label or identifier of a content item can be constructed by walking the tree from the root, counting the preceding siblings at each level. Thus the root node, which can have no siblings, is always the first node at the top level, identified as “1”. The first child of the root node is then identified as “1.1”. The fourth child of the first child of the root node is “1.1.4”. And so on.

Notice that numbering begins from one, not zero, and that the root node is always included.<sup>34</sup> Figure 21 illustrates the same example as shown in Figure 20, with each node annotated with its implicit label.

The reference mechanism does not permit references to nodes within other documents. That is, one can only have “by reference” relationships with content items

---

32. Indeed it was the requirement for this “side effect” that was the reason for this restriction in the first place.

33. The Unsigned Long (UL) value representation in DICOM is a 32 bit unsigned binary word.

34. Even though inclusion of the root node is redundant, that is, all references always begin with “1”. This redundancy is necessary however, in order to be able to reference the root node itself.

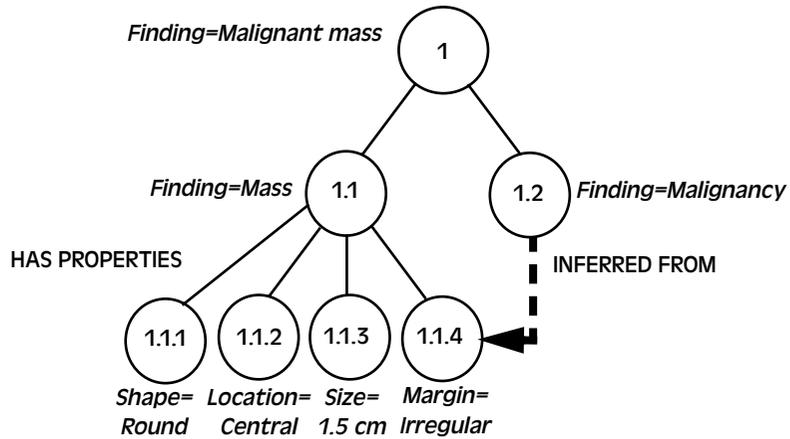


FIGURE 21. Implicit Numbering of Nodes for References

within the same document. One can refer to entire documents by using the COMPOSITE value type, but there is no provision to reference individual items within that document. Conceptually, this would be possible by a combination of the SOP Instance UID of the target document and a referenced content item. However, it is forbidden as a matter of policy, on the grounds that the “internal arrangement” of a referenced document might change, outside the control of the creator of a referencing document.<sup>35</sup>

### Selected From Relationship Type

The “selected from” relationship specifies that the parent (source node) is a set of coordinates selected from a child (target node) that is either another set of coordinates, or an image or waveform.

The use of the “selected from” relationship was briefly mentioned in the chapter on “Content Items: Concept Names and Values” in the section on coordinates. It will be explained in greater detail in the chapter on “Images, Waveforms, and Voice Audio.”

### Has Observation Context Relationship Type

The “has observation context” relationship extends the observation context of a parent (source node), and all its descendants, with the additional context specified in a child (target node).

35. Although it is fair to say that the rules for the SR SOP classes in PS 3.4 forbid such changes without assignment of a new SOP Instance UID.

The whole subject of observation context is complex, sufficiently so to deserve an entire chapter of its own (see “Context: Who did what to whom, when...”). The use of the “has observation context” relationship will be explained there.

## Has Acquisition Context Relationship Type

The “has acquisition context” relationship extends the acquisition context of a parent (source node) with the additional context specified in a child (target node). Unlike the “has observation context” relationship, there is no explicit definition that this additional context propagates to the other descendants of the parent (inheritance).

Acquisition context is described in the chapter on “Images, Waveforms, and Voice Audio.”

## Has Concept Modifier Relationship Type

The “has concept modifier” relationship modifies the meaning of the concept name of a parent (source node) with the modifier or qualifier specified in a child (target node).

This relationship allows for the possibility of:<sup>36</sup>

- ▶ constructing post-coordinated concepts
- ▶ specifying code meanings longer than 64 character limit of the value representation
- ▶ specifying one or more free text or coded language translations

It is important to notice that the modifier applies to the *concept name* of the parent, not to the value, and in particular not to a value that is a coded entry.

The difference between a pre-coordinated and a post-coordinated concept will be illustrated by way of an example. Suppose one wants to define the concept of an erect, single view, postero-anterior (PA) chest X-ray. This is such a common radiological procedure that it can be considered as a concept in its own right. One could imagine a code in a coding scheme that applied to the entire concept “erect, single view, PA chest X-ray”. This is a “pre-coordinated” definition.

On the other hand, one could consider this as just an instance of an X-ray in general.<sup>37</sup> In this case, one might *qualify* or *modify* the basic concept by adding further descriptors, such as the fact that a single projection or view is to be taken, that the projection or view is to be postero-anterior, and that the position of the patient with respect to gravity is to be erect.<sup>38</sup> This would be a “post-coordinated” definition.

---

36. No doubt implementers will conceive of other uses (and abuses) of this relationship.

37. Using “X-ray” carelessly as a synonym for “projection radiograph” in this example, as is common parlance.

38. Those not familiar with radiology may wonder why position with respect to gravity matters. Not only does an erect film give a better view of the lung bases, but also any fluid will gravitate to dependent spaces. The appearance of a supine film is quite different.

One can either make use of pre-coordinated codes such as:

```
<CODE:(,,,"Exam")=(,,,"Erect, single view, PA chest X-ray")>
```

or post-coordinated concepts, by using the “has concept modifier” relationship, such as:

```
<CODE:(,,,"Exam")=(,,,"Chest X-ray")>
  <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"# of views")=(,,,"Single")
  <has concept mod CODE:(,,,"View")=(,,,"PA")>
```

Why is this distinction useful? There are at least two reasons:

- ▶ extensive use of pre-coordinated codes results in a huge expansion of the number of codes required, essentially one for every plausible combination of more atomic concepts
- ▶ separation of concepts into atomic components makes searching for individual features much easier

Despite the large number of codes required, pre-coordinated codes are still commonly used in many applications. A special case of pre-coordinated codes are units of measurement, which can much better be described and coded as a “composite” of the basic physical units from which they are comprised. In reality however, one can expect to be using single pre-coordinated codes for units for the some time to come. For example, a single code that means “milliliter/second” will likely be preferred over a code constructed from the concepts of “milliliter”, “division” and “second.”<sup>39</sup>

As a (rather contrived) example of how post-coordinated descriptions make searching easier, consider how one might search for all X-rays taken with the patient erect, regardless of body part, from amongst the following cases:

```
<CODE:(,,,"Exam")=(,,,"Erect, PA chest X-ray")>
<CODE:(,,,"Exam")=(,,,"Erect and supine abdominal X-ray")>
<CODE:(,,,"Exam")=(,,,"Routine facial fracture screen")>
<CODE:(,,,"Exam")=(,,,"Trauma lateral c-spine X-ray")>
```

When using pre-coordinated codes like those above, the search application just has to magically “know”<sup>40</sup> that the codes for the chest and abdomen include views in the

39. See the “Unified Code of Units of Measurement” at “<http://aurora.rg.iupui.edu/UCUM/>” for further discussion of these issues.

Pre-coordinated codes for other than atomic units can be “constructed” according to the UCUM rules. For example a pre-coordinated code value for “milliliter/second” could be constructed as “ml/s” rather than defining an “arbitrary” code like “12345”. However, there is controversy over this approach. Some coding schemes (such as SNOMED) deprecate as a matter of policy the use of code values that “have meaning” and can be rendered or parsed.

40. The “magic” referred to in this case is not intractable, however. If one has a “semantic network” that describes how pre-coordinated codes are related to one another, then it is possible to perform the sort of search described. It is the definition of these sorts of relationships between codes that distinguishes coding schemes like SNOMED-RT from earlier versions that were simple lists of coded concepts.

erect position, as does the facial fracture study, but that the lateral c-spine for trauma does not.

Were these to be described in a post-coordinated manner however, using consistent modifiers, the search application would only have to look for the single code that indicated that the patient was in the erect position:

```
<CODE:(,,,"Exam")=(,,,"Chest X-ray")>
  <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"View")=(,,,"PA")
<CODE:(,,,"Exam")=(,,,"Abdominal X-ray")>
  <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"Position")=(,,,"Recumbent")
  <has concept mod CODE:(,,,"View")=(,,,"AP")
<CODE:(,,,"Exam")=(,,,"Facial bones X-ray")>
  <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"View")=(,,,"Lateral")
  <has concept mod CODE:(,,,"View")=(,,,"Waters")
  <has concept mod CODE:(,,,"View")=(,,,"SMV-tangential")
<CODE:(,,,"Exam")=(,,,"C-spine X-ray")>
  <has concept mod CODE:(,,,"Position")=(,,,"Recumbent")
  <has concept mod CODE:(,,,"View")=(,,,"Lateral")
  <has concept mod CODE:(,,,"Special")=(,,,"Trauma")
```

This example also illustrates the importance of using modifiers consistently, and having a good repertoire of them from which to choose. In the real world, one would need more precisely defined codes than those used here.<sup>41</sup>

One would also need the ability to construct more complex structures involving multiple modifiers. Consider, for example, how to describe the three-view erect and supine abdomen and erect chest examination that is often performed in the investigation of the “acute abdomen.”<sup>42</sup> This involves describing multiple dimensions of multiple views. One approach might be encoded as:

```
<CODE:(,,,"Exam")=(,,,"Projection X-ray")>
  <has concept mod CODE:(,,,"Region")=(,,,"Chest")
    <has concept mod CODE:(,,,"View")=(,,,"PA")
    <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"Region")=(,,,"Abdomen")
    <has concept mod CODE:(,,,"View")=(,,,"PA")
    <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <has concept mod CODE:(,,,"Region")=(,,,"Abdomen")
    <has concept mod CODE:(,,,"View")=(,,,"AP")
    <has concept mod CODE:(,,,"Position")=(,,,"Recumbent")
```

41. “View” and “position” for example, are used loosely without precise definition.

42. This traditional procedure is designed to detect thoracic abnormalities masquerading as painful abdominal conditions, free gas underneath the diaphragm as a result of perforated viscera and the pattern of fluid distribution in the bowel. Hence the need for erect and supine positioning of the patient.

This is not quite right however, since the “view” and “position” are not really modifiers of the anatomical “region” per se. Perhaps a better approach<sup>43</sup> would be:

```
<CONTAINER:(,,,"Exam")>
  <contains CODE:(,,,"Image")=(,,,"Projection X-ray")
    <has concept mod CODE:(,,,"Region")=(,,,"Chest")
    <has concept mod CODE:(,,,"View")=(,,,"PA")
    <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <contains CODE:(,,,"Image")=(,,,"Projection X-ray")
    <has concept mod CODE:(,,,"Region")=(,,,"Abdomen")
    <has concept mod CODE:(,,,"View")=(,,,"PA")
    <has concept mod CODE:(,,,"Position")=(,,,"Erect")
  <contains CODE:(,,,"Image")=(,,,"Projection X-ray")
    <has concept mod CODE:(,,,"Region")=(,,,"Abdomen")
    <has concept mod CODE:(,,,"View")=(,,,"AP")
    <has concept mod CODE:(,,,"Position")=(,,,"Recumbent")
```

In this example, a container value type has been used to model the description of the exam. This may not be possible for some SOP classes. The “has concept modifier” relationship is constrained to a lesser extent than other relationships in the simpler SOP classes.<sup>44</sup> Accordingly, it may be necessary to use “has concept modifier” to construct more complex post-coordinated concepts, even though “contains” or “has properties” relationships might be more meaningful.

Regardless, these examples hint at how the “expressiveness” of the SR tree can be used to construct complex post-coordinated concepts.<sup>45</sup>

Other applications of the “has concept modifier” relationship arise as a consequence of the need to transmit various different meanings for coded entries. In the chapter on “Codes and Controlled Terminology” it is emphasized (ad nauseam) that a receiving application should look up the code value and coding scheme designator in its own lexicon to find an appropriate meaning to render. Were all applications to do so, they could use appropriate meanings of whatever length were available, in whatever local language was appropriate. However, the code meaning attribute is mandatory for various pragmatic reasons. Not all receivers will have the most recent version of the dic-

43. There is no intent here to imply that the concepts shown correspond to the various entities within the DICOM information model, such as studies, procedure steps, action items, protocols, series or instances. The term “exam” as used here corresponds loosely to either a “study” or a “performed procedure step” in DICOM, depending on how the work is organized.

44. Specifically, both the Basic Text and Enhanced SR SOP classes forbid the use of code value types as children of other code value types except “has concept modifier.”

45. Earlier proposals to attach one level of modifier attributes to coded entries were rejected in favor of re-using the capabilities of the tree. This led to the introduction of the “has concept modifier” relationship, which was already being considered as a solution to the restricted code meaning length problem. The need became apparent as use cases were derived to justify having multiple lists of modifiers, and then ordinal correspondence between such lists, and so on. As the requirements for expressing post-coordinated concepts became more sophisticated, it was obvious that the SR tree could be used to achieve the same effect without introducing entirely new encoding mechanisms.

tionary. Even if they do, without consideration of the context in which a code is used, it might not be easy to choose the most appropriate synonym to render.

Given the need to sometimes directly render the code meaning, the 64 character limitation of the value representation used to encode it presents a problem. Many pre-coordinated codes require a lot of text to describe their full meaning, and only hopelessly cryptic abbreviations will fit into 64 characters. While this is yet another good reason to make more extensive use of post-coordinated codes, that approach is not realistic in all situations. Codes for laboratory and other diagnostic procedures are typically in this class.<sup>46</sup> An example from the ICD10-PCS coding scheme for procedures illustrates the problem:

```
CODE: (209007,99PMP,"Procedure")=(50291CC,I10P,
    "IMAGING:CNS:CT:SELLA:LOWOSMOLAR:IT,U,E:2PLANE3D")
```

In this example, a relatively complex procedure is concisely and uniquely identified by the code value "50291CC". However, it is rather hard to construct a "meaningful" code meaning that fits in 64 characters.<sup>47</sup> Applications that want to transmit a more meaningful string (to render when the lexicon is not available) can use the "has concept modifier" relationship, as follows:

```
CODE: (209007,99PMP,"Procedure")=(50291CC,I10P,
    "IMAGING:CNS:CT:SELLA:LOWOSMOLAR:IT,U,E:2PLANE3D")
<has concept mod TEXT:
    (209600,99PMP,"Equivalent meaning of value")=
    "CT imaging study of the sella turcica/pituitary
    gland, unenhanced, with intravenous contrast, and
    with intrathecal contrast, using low osmolar contrast,
    in two planes, with 3D reconstructions">
```

Whether this is actually "better" or "more useful" or not is a matter of opinion, but the mechanism is available.

Success of this approach depends on the definition of appropriate codes for concepts of the target of the concept modifier relationship, and their widespread adoption. That is, everyone will have to agree to use the same code for "Equivalent meaning of value" and to use the same template.

The need to transmit one or more localized language translations of coded entries can be addressed in a similar manner. This will be exhaustively discussed in the chapter on "Internationalization and Localization."

One aspect of concept modifiers that remains slightly ambiguous is whether or not multiple concept modifiers have an "and" or an "or" relationship with each other or, for that matter, with their parent. The intent is that multiple concept modifiers that apply to a single node are cumulative, so that all modifiers apply. Whether or not a

46. Codes from the widely used LOINC classification are also good examples of this.

47. Nor does ICD10-PCS actually specify a mechanism for constructing such meanings; what is shown here is the author's own best effort.



particular modifier specifies an alternative for, or a constraint upon, the meaning of the parent concept depends upon the concept of the modifier.

## Summary

Some of the key points discussed in this chapter include:

- ▶ structured reports consist of trees rather than flat lists
- ▶ the tree is encoded as recursively nested sequence items
- ▶ each sequence item is a “content item” or “node,” with parents, more distant ancestors, siblings, children and more remote descendants
- ▶ the nesting is documented in the standard by recursive invocations of macros, specifically the Document Content and Relationship Macros in the SR Document Content Module
- ▶ unlike other structured document standards like XML, DICOM SR specifies explicit relationships between parents and children; these are mandatory
- ▶ relationships are encoded within the child, so that a single parent can have different relationships with different children
- ▶ the “contains” relationship specifies containment; that is the children are part of (contained within) the parent
- ▶ the “contains” relationship is often used to specify an outline or hierarchy of headings; the concept names of containers serve as headings of sections, and their contained sub-sections in turn have concept names that are sub-headings
- ▶ only “container” value types can “contain” content (i.e. can use the “contains” relationship)
- ▶ the “has properties” relationship is used to describe properties, attributes or characteristics of a content item
- ▶ the “inferred from” relationship is used to described from what information a content item was derived
- ▶ the “selected from” relationship is used to apply coordinates to an image or waveform reference
- ▶ the “has observation context” and “has acquisition context” relationships are used to attach context to a content item
- ▶ the “has concept modifier” relationship is used to construct post-coordinated concepts, to specify longer meanings, and to specify alternative meanings such as in localized languages
- ▶ relationships “by-reference” allow construction of a more complex data structure than a tree
- ▶ containment is not permitted across “by-reference” relationships
- ▶ references are to implicitly numbered, ordered nodes, along a path descending from the root, rather than to explicit labels
- ▶ references can only be to content items within the same document, not to content items in external documents

# ***Images, Waveforms, and Voice Audio***

---

A picture is worth a thousand words, or so the saying goes. In the medical imaging field, pictures are the input, but reports are the output in most cases. The value of a diagnostic procedure lies more in the interpreting physician's expert opinion than the source images themselves. Traditional reports typed on paper are self-contained and meaningful in their own right, but are most useful when reviewed in conjunction with the images. Frequently, plain films are annotated with wax-pencil graffiti to highlight areas of significance to the reader of the report. As softcopy images proliferate and reports migrate on-line, the link between details in the report and areas of interest in particular images becomes even more crucial. Indeed, in implementations that lack such a link, the wax pencil is sorely missed. DICOM SR exists primarily to address this need.

## **Report+Images**

The state of the art in PACS<sup>1</sup> and information systems that attempt to combine reports and images, is the so-called "multi-media" report. Such a report consists of a "report plus images."<sup>2</sup> Rarely, if ever, is there support for explicit links between findings in the report and specific features within images. DICOM SR addresses this deficiency by providing such links. Nevertheless, the value of providing any kind of access to both the report and the images at the same time should not be underestimated. Many softcopy work environments maintain reports and images completely separately. This requires studies to be selected manually on two different terminals (or sometimes three if the dictation system requires its own identifier entry).<sup>3</sup> In many cases, not even old reports are available on the imaging workstations.

---

1. *Picture Archiving and Communications Systems.*

2. *This distinction was emphasized by Dean Bidgood in his early presentations on the subject of SR.*

## The Limits of the Web Paradigm

No doubt everyone reading this book will have used a web browser. It is no secret that web pages are easy to generate and may contain references to images. These images are displayed “in line,” or sent to a plug-in or a spawned external application. This may seem like an attractive approach for reviewing diagnostic imaging reports. However, the web-based approach suffers from several key flaws:

- ▶ the power of the HTML page to directly embed images is limited to relatively low quality pre-formatted images<sup>4</sup>
- ▶ there is no facility to link text on the HTML page to features within an image (only to entire images)
- ▶ spawning an external viewer to handle full fidelity images may lose the link between a report feature and the image referenced
- ▶ there is no structured or coded content to index, query or analyze<sup>5</sup>

A simplistic web approach, though a considerable improvement, is limited to a “text+images” multimedia report.

To do better, one needs to either:

- ▶ convey structured content to the client (browser) side
- ▶ build much more complicated pre-formatted content on the server side (for example, pre-formatted images that contain burned in annotations)
- ▶ make the client-server interaction more sophisticated than a simple http transaction (for example, by using applets to communicate directly with the server and then render more complex content)

These approaches are all feasible, but still require a source of structured content. The unassisted browser is revealed as no more than a highly portable graphical user interface, which can be used to present the output of a more complex application that is really doing the hard work.<sup>6</sup>

The question remains as to where the structured content comes from, and how it is interchanged. In order to be able to search or query a report for features or codes, the interchange format should contain content that is structured rather than pre-formatted. If every report generating application created only pre-formatted web pages,<sup>7</sup> they might be universally “viewable,” but of little use for more sophisticated analysis.

---

3. Even if the image review and reporting applications are running on the same workstation or desktop, they may not be integrated or even “aware of each other.” Hopefully, shared context standards such as CCOW will be adopted to address this problem.

4. Specifically, support in most browsers is limited to indexed color GIF images and lossy compressed true color JPEG images. More widespread support for the <EMBED> tag may alleviate this problem in the future.

5. Though future XML-aware browsers may have greater capability in this respect.

6. Or “executing the business logic” in contemporary parlance.

7. Or for that matter, any other form of pre-formatted content such as Postscript, PDF, RTF, etc.

They would be little better than the current plain text report. Increasingly, organizations are expected to generate information rather than just data. A vast amount of information is embedded in traditional plain text reports, but is irretrievable for all practical intents and purposes, to any “data mining” application.<sup>8</sup>

## Image References

DICOM SR provides features to:

- ▶ reference individual images (and frames within multi-frame images)<sup>9</sup>
- ▶ specify a presentation state<sup>10</sup> to be applied to the referenced images
- ▶ reference other DICOM objects such as time-based waveforms (EKGs, etc.)
- ▶ specify spatial coordinates within an image, such as the outline of a mass, or the endpoints of a linear measurement
- ▶ specify temporal coordinates within an image or waveform, such as the end of systole or diastole on a cardiac image or waveform
- ▶ specify the purpose of the reference to the coordinate, image or waveform

These features are the core of what DICOM SR provides over and above the simple multi-media report.

In the chapter on “Content Items: Concept Names and Values,” the encoding of the IMAGE, SCOOD and TCOORD value types was described. The role of the concept name in conveying the purpose of reference was introduced. How spatial coordinates may be defined as single or multiple points, polylines or ellipses was illustrated. In the chapter on “Encoding & Relationships: Building the Tree,” the SELECTED FROM relationship was introduced, but further explanation was deferred until now.

This section will describe how image references, coordinates and relationships may be combined in a structured report. A later section in this chapter will address how waveforms may also be included.

The simplest pattern is to include a reference to an entire image in a report. The image content item itself might be placed within a container. For example, one could encode:

```
<CONTAINER:(209076,99PMP,“Chest X-Ray Report”)>  
  <contains CONTAINER:(209048,99PMP,“Images”)>  
    <contains IMAGE:=(DX Image,“1.2.3.4”)>
```

- 
8. *This is perhaps overstating the case a little. As any regular user of Internet search engines can attest to, plain text searches are still quite useful for ad hoc, unsophisticated queries.*
  9. *The question of conformance and support of references to composite object types is addressed in the chapter on “SOP Classes and Conformance.”*
  10. *Grayscale Softcopy Presentation State Storage objects store geometric and contrast transformations to apply to an image in order to reproduce the same appearance on different devices. A magnified and windowed CT slice, for example, should look the same, or very similar, wherever it is viewed.*

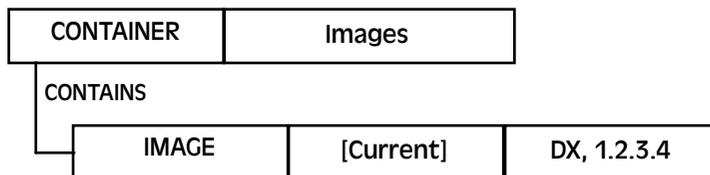


FIGURE 22. Simple Image Reference Pattern ± Purpose

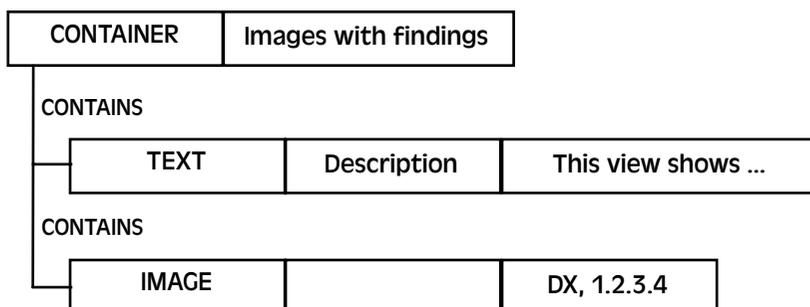


FIGURE 23. Key Image Note Image Reference Pattern (IHE Y2)

The referenced image in this case is a single frame object, and hence individual frames are not specified. Nor has a presentation state been included.

In this example, there is no indication what the purpose of the reference to the image is.<sup>11</sup> The purpose of the reference could be conveyed in the concept name attribute of the image reference itself, as follows (see Figure 22):

```
<CONTAINER:(209076,99PMP,"Chest X-Ray Report")>
  <contains CONTAINER:(209048,99PMP,"Images")>
    <contains IMAGE:(209047,99PMP,"Current")=
      (DX Image,"1.2.3.4")>
```

This simple pattern of references to images is useful, but just as limited as the web-based multimedia approach that was described earlier. One potential application of a pattern like this is to convey in a single object a “key image note” as defined by the IHE in their Year 2 framework (see Figure 23). In that pattern, the entire document consists of a single image reference and a text description, optionally with some observation content. For example:<sup>12</sup>

11. Recall that the concept name is optional for the IMAGE value type, as it is for WAVEFORM, SCORD and TCOORD.

12. Private codes are used in this example for the document title, the image purpose of reference and the concept name of the descriptive text. At the time of writing, the IHE technical committee had not yet suggested values for these concepts.

```
<CONTAINER:(209046,99PMP,"Images with findings")>
  <contains IMAGE:=(DX Image,"1.2.3.4")>
  <contains TEXT:(209045,99PMP,"Key image description")
    ="This view shows the abnormality">
```

The “key image note” pattern was introduced to address the common complaint that it is very difficult to “flag” particular images in a study as being of interest.

## Presentation States

A grayscale softcopy presentation state may be encoded with an image reference. Presentation states have three primary roles, all of which are aimed at achieving consistency of appearance, regardless of the display device. Presentation states specify:

- ▶ a fully defined grayscale contrast transformation pipeline from the source pixel data to a standard grayscale space
- ▶ spatial transformations, such as selection of an area to be displayed and how the selected area should be rotated and/or magnified
- ▶ vector graphic annotations that may be applied

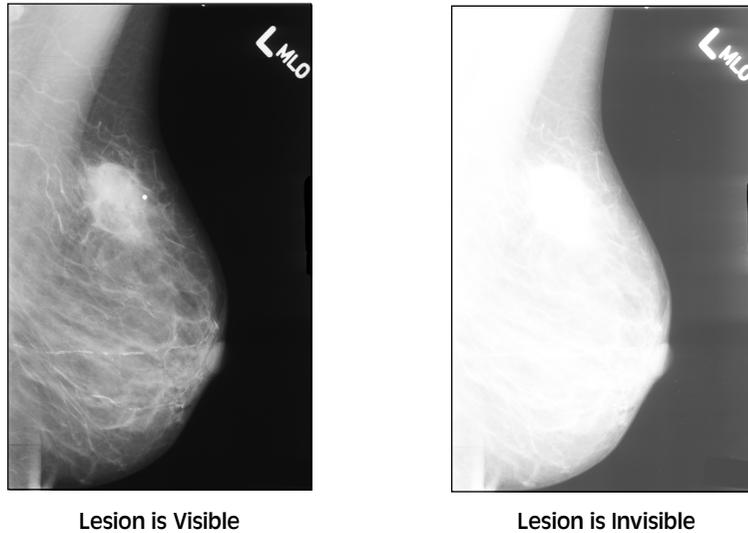
Medical images are often stored in a form that is not able to be directly displayed. For example, a CT image is usually stored with greater than eight bit precision, often in a signed integer form corresponding to the density in Hounsfield Units.<sup>13</sup> In order to view or print such images satisfactorily, a range of the available pixel values needs to be selected and mapped to the full range of the display or print device. In the case of CT, this is usually a linear operation and is referred to as “windowing,” or choosing a “window center and width.” Different choices of window may be applied to the same image alternately, in order to better illustrate different anatomical or pathological features. Hence terms like “soft-tissue windows” and “bone windows” are used. Images from other modalities are also often greater than eight bits deep and require grayscale contrast transformation. These include MR, computed radiography and digital X-ray images. Contrast transformations are not always expressed as a “window” operation and may be perceived by the operator as “brightness” and “contrast” adjustments. The transformations are not always linear, and may involve correction for the characteristics of the source and display or print devices. Compensating for a display device’s characteristic curve<sup>14</sup> is also necessary, even for eight bit images.

Images that are displayed on low cost personal computer monitors (in conjunction with reports) are often not appropriately adjusted for the characteristics of the display, nor for the characteristics of the source. This problem is exacerbated when images are displayed in-line by a web browser. One of the major advantages of referencing DICOM images in a report is that they may be conveyed in their full fidelity to the review system. The DICOM image may then be adjusted:

---

13. Hounsfield Units are a measure of density. Water is defined as zero; fat and air have negative values; soft tissue, contrast media and bone have positive values.

14. Sometimes approximated by correcting for the display system’s “gamma.”



**FIGURE 24. Effect of Contrast Transformations**

- ▶ by the user, to their preference, given appropriate controls
- ▶ by the application, to compensate for the display device and ambient lighting conditions, given appropriate calibration mechanisms
- ▶ by automatic application of a referenced DICOM presentation state, to highlight predetermined features of interest

This is not to say that a high-quality, calibrated, grayscale monitor in a darkened room is required at every location where a structured report with images is viewed. Rather, it is to highlight that the information to make the appropriate adjustments can be made available and used as appropriate. Consistency of grayscale appearance on different devices requires consensus on a “standard” grayscale curve. In DICOM, such a curve is defined by using the concept of “perceptual linearization.” This means that an equal change in input will result in an equal perceived change in contrast by a human observer, regardless of the absolute luminance. That is, regardless of whether the change is in the light, mid-gray or dark region of the luminance range, and regardless of the ambient light.

The system architecture for the distribution of reports with images may support either:

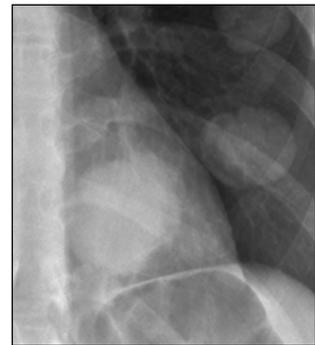
- ▶ distribution of “original” DICOM images with presentation states to “client-side” applications that will combine them to render them appropriately for the display
- ▶ “server-side” rendering of images and presentations states into a form suitable for distribution as eight bit “pre-formatted” images appropriate to the target display device



Original is wrong way around



Apply horizontal flip to correct orientation



Show retrocardiac mass by zoom/crop/adjust contrast

**FIGURE 25. Spatial and Contrast Transformations**

Either approach is equally valid. Regardless, even when there is little control over the type of display or the viewing environment, a modest effort at compensating for the “generic” characteristics of low cost color monitors on personal computers makes a great difference. For example, just compensating for the different “default gamma” choices on PC, Mac and Sun operating systems has a dramatic (beneficial) effect.

In a medical imaging department, the workflow is usually organized such that expert radiographers and technologists adjust images prior to distribution such that they are suitable for viewing. In a film-based environment, such adjustments typically involve choosing the correct grayscale transformations, as well as spatial transformations and user annotation, prior to printing. For example, a direct coronal CT examination of the paranasal sinuses often involves not only choosing the correct window center and width, but also rotating and flipping the images from the default “upside down” presentation, as well as annotating them to indicate which side is really left or right.

In a soft-copy work environment, the same transformations can still be performed by the operators, then stored as presentation states. This saves the end user, such as the radiologist or referring physician, from having to repeat the same task, one which is very tedious and irritating for them. The alternative of saving a pre-formatted image with the transformations “burned in” is less flexible, since it prevents any later adjustment of the image parameters. For example, the “default” presentation state for a paranasal sinus CT examination generated by an operator (or automatically by the system) may reveal an abnormality. When the radiologist or surgeon sees the abnormality, they may wish to magnify the image and adjust the window in order to determine if there is bony involvement, and so on.

Presentation states saved with images and referenced in structured reports have the potential to greatly enhance the acceptability of the output rendered to the end-user. Even when using default window settings and default assumptions about low-cost target display systems without calibration, the results are preferable to ignoring the issue entirely.

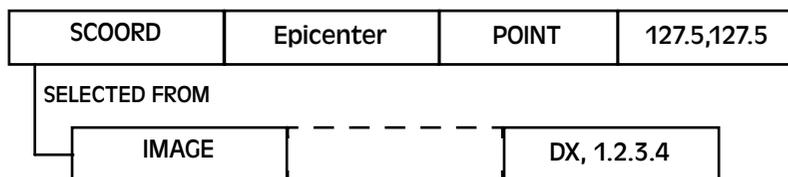


FIGURE 26. Spatial Coordinates Selected From an Image

DICOM presentation states contain mechanisms for adding user annotations in the form of text and vector graphics. These are purely presentation-oriented features, with no semantics attached. While it may sometimes be necessary to attach such annotations to images at the acquisition device, it is preferable to use the SR mechanisms of spatial coordinates to provide semantically meaningful “mark-up” of an image.

### Spatial Coordinates

A wax pencil is the tool traditionally used by radiologist to annotate a plain film or printed image. Marks made with a wax pencil are not indelible and may be removed to reveal underlying features or for film duplication. The marks are durable though, so they are not inadvertently lost. One of the major limits of the wax-pencil “paradigm” is that there is no direct correspondence between comments in the text report and features marked on the film. One may encounter reports like “the 2mm lesion is shown circled in the top right hand corner of slice 23 of the second echo series 6 and a corresponding abnormality is present in the same location on slice 17 of the post-contrast series 8.” In a soft-copy world with structured reporting one can do much better, through the use of spatial coordinates.

Spatial coordinates are defined in their own content item, with a value type of SCOORD. A special relationship, SELECTED FROM, is to use to specify the image content item from which the coordinates are selected. This is encoded as shown in the following example and illustrated Figure 26:

```
<SCOORD:(209239,99PMP,“Epicenter”)=(POINT,127.5,127.5)>
  <selected from IMAGE:=(DX Image,“1.2.3.4”)>
```

Notice that the coordinates are encoded in a separate node of the tree from the image reference itself. There are several reasons for this. First, separating the coordinates from the image reference allows a single coordinate reference to be applied to multiple images.<sup>15</sup> For example, one might specify a cylindrical volume of interest through a block of contiguous parallel CT slices in the following manner:

15. An error in Supplement 23 suggested that references to multiple images could be contained in a single IMAGE value type. A proposal to correct this error is contained in CP 216. The mechanism intended for applying the same coordinates to multiple images is to use multiple IMAGE value types.

```
<SCoord:(209240,99PMP,"Selected volume")=
  (CIRCLE,127.5,255.5,145.5,255.5)>
  <selected from IMAGE:=(CT Image,UID of image 1)>
  <selected from IMAGE:=(CT Image,UID of image 2)>
  ...
  <selected from IMAGE:=(CT Image,UID of image n)>
```

This works because the coordinates are always specified in a two-dimensional image-relative space, not a patient- or device-relative three-dimensional space. Remember also that the coordinates are in an image-relative space that is defined *before* application of presentation state. That is, the coordinates are the same regardless of whether a presentation state is applied or not, and whether or not it specifies any spatial transformations such as rotation, flipping or magnification.

Another reason for separating the coordinates into a separate node from the image reference is to allow both spatial and temporal coordinates to be selected from the same image. This will be described in detail later in this chapter.

The purpose of reference is optional for spatial coordinates and images.<sup>16</sup> The intent is that when an image reference occurs without any coordinates, then a purpose of reference may be sent as the concept name of the image content item, indeed it probably should. When coordinates are selected from an image, however, it makes more sense to denote the purpose of reference with the coordinate node. This is because the image reference itself has no “purpose” per se, other than to act as the target of the coordinates. The “concept” being identified is defined by the coordinates applied to the image, not the image itself. The corollary of this is that images that are the targets of SELECTED FROM relationships with coordinates should never have a concept name.<sup>17</sup>

## Temporal Coordinates

Temporal coordinates, like spatial coordinates, also occur in separate nodes from the references to the objects they are selected from. The same SELECTED FROM relationship is used. Unlike spatial coordinates which can *only* be selected from images, temporal coordinates can be selected from:

- ▶ images
- ▶ waveforms
- ▶ spatial coordinates

Since waveforms do not have two spatial dimensions, spatial coordinates can never be selected from them. The simplest form of temporal coordinate reference is therefore a TCOORD content item selected from a WAVEFORM content item, as in the following example:

---

16. For that matter, the purpose of reference is also optional for temporal coordinates, waveforms and composite object references.

17. It is not actually forbidden by the standard, it just doesn't make sense.

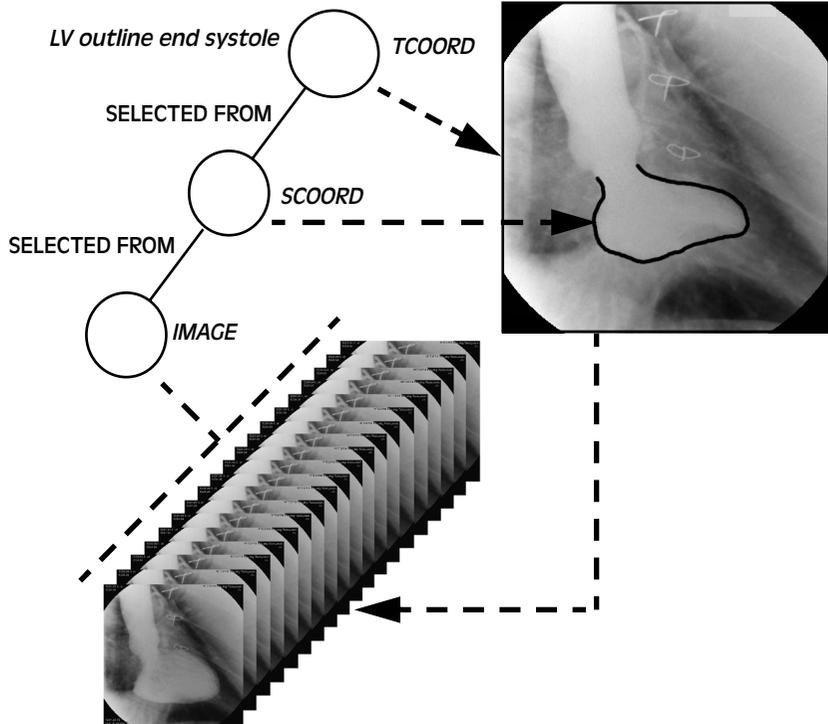


FIGURE 27. Temporal and Spatial Coordinates with an Image

```
<TCOORD:(109041,DCM,"P wave")=(POINT,0.310 secs)>
  <selected from WAVEFORM:=(12 Lead EKG,"1.2.3.4")>
```

A more interesting pattern occurs when a single temporal coordinate item is applied to both waveforms and images, as in the following example:

```
<TCOORD:(209801,99PMP,"End systole")=(POINT,0.310 secs)>
  <selected from WAVEFORM:=(Hemodynamic,"1.2.3.4")>
  <selected from IMAGE:=(XA Image,"1.2.3.5")>
```

The specific details of the various types of temporal coordinate that may be used are described in the chapter on "Content Items: Concept Names and Values." There it is pointed out that one type of temporal coordinate, sample position, applies only to waveforms. Therefore, if a single coordinate content item is to apply to both an image and a waveform,<sup>18</sup> only the absolute time or relative temporal offset coordinates may be used.

For a temporal reference to an image to make sense, the image has to be of a multi-frame image SOP class. Furthermore, there should be no selection of a specific frame

18. It is permitted for there to be more than one image content item and/or more than one waveform content item as the children of a single temporal coordinate content item.

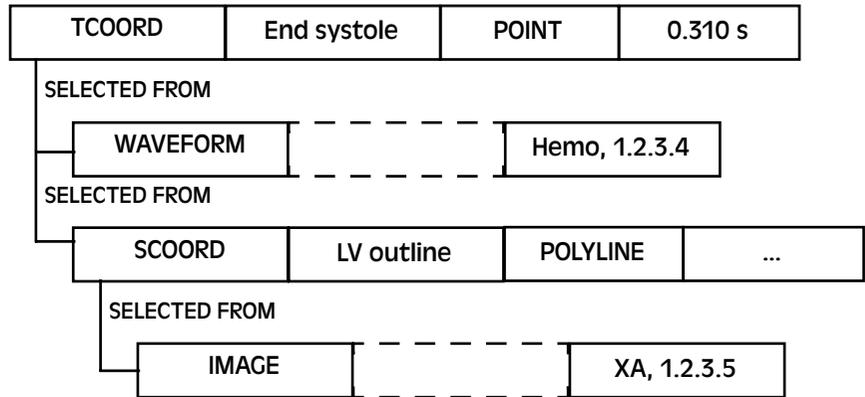


FIGURE 28. Temporal and Spatial Coordinates

number in the image reference itself, but rather the entire image should be referenced.

The most interesting patterns occur when there is both temporal *and* a spatial coordinate selection. The rules for the coordinate value type macros define the organization of this pattern in the tree.<sup>19</sup> Specifically, the temporal coordinate reference always occurs highest in the tree, followed by the spatial coordinate reference, then the image reference which occurs lowest in the tree. It is often appropriate to attach the purpose of reference only to the highest node. The following example (illustrated in Figure 27) demonstrates the order:

```
<TCOORD:(209799,99PMP,"LV outline at end systole")=
(PPOINT,0.310 secs)>
  <selected from SCOORD:=(POLYLINE,...)>
    <selected from IMAGE:=(XA Image,"1.2.3.5")>
```

A slightly more complex example (shown graphically in Figure 28) illustrates how the temporal coordinates may be selected from waveforms in addition to spatial coordinates from images:

```
<TCOORD:(209801,99PMP,"End systole")=(POINT,0.310 secs)>
  <selected from WAVEFORM:=(Hemodynamic,"1.2.3.4")>
  <selected from SCOORD:(209800,99PMP,"LV outline")=
(POLYLINE,...)>
    <selected from IMAGE:=(XA Image,"1.2.3.5")>
```

The last example illustrates one approach to defining appropriate purposes of reference at the different levels of the pattern. Since the temporal coordinates apply to both the waveform and the image, the time offset that is the end of systole is separated from the spatial coordinates that are the outline of the left ventricular volume.

19. That is, the restrictions are fundamental to the SR framework, and are not specific to any particular SOP class or template.

## Waveforms

Composite objects that encode waveforms have been mentioned already, in terms of how they may be referenced using the WAVEFORM value type and how they may be the target of a temporal coordinate selection.

Waveforms are supported in DICOM by a family of time-based waveform storage SOP classes. At one time it was proposed that there should be a single generic waveform SOP class, capable of encoding any form of time-based waveform, no matter what number of channels were sampled, how often, with what precision or for what purpose. That is, the SOP class would be a generic container for waveforms.<sup>20</sup> However, this would create problems for applications such as the exchange of voice audio, or traditional 12-lead electrocardiograms (EKGs). For these applications, receiving devices might have difficulty if the form of the waveform data was not tightly constrained to within reasonable limits. Furthermore, consensus on the descriptive attributes for specific applications is required for interoperability. So, just as for images, a family of highly specific SOP classes were defined for time-based waveforms. All of these specific SOP classes share a common encoding and framework, but differ in the details. There is no generic waveform SOP class.<sup>21</sup>

TABLE 4. Waveform IODs and Storage SOP Classes

Basic Voice Audio
12-Lead Electrocardiogram
General Electrocardiogram
Ambulatory Electrocardiogram
Hemodynamic
Basic Cardiac Electrophysiology

It is beyond the scope of this book to go into detail about how waveforms are encoded in DICOM. In summary though:

- ▶ only time-based waveforms are supported<sup>22</sup>
- ▶ a collection of samples acquired with the same sampling rate over a period of time is called a “multiplex group”
- ▶ within each multiplex group there may be multiple “channels”
- ▶ a “temporal frame of reference” can be defined, such that multiple objects that share the same temporal frame of reference can be synchronized; this synchroni-

20. This is the approach taken by some other standards like HL7 V2.3, which provide the framework to describe a waveform, without constraining its content.

21. The intent is that as new applications arise, new specific SOP classes can rapidly be defined using the common framework. For example, the ultrasound working group has proposed a new high frequency audio SOP class intended for encoding Doppler audio signals. See Supplement 56.

22. This is distinct from the old, rarely used, curve objects in DICOM. Curve objects use similar mechanisms to encode time based waveforms and multi-dimensional curves and vector graphics.

zation may be relative or with respect to absolute time; the objects can be waveforms or images or both

- ▶ clinical and modality specific acquisition technique information is not encoded in traditional DICOM data elements, but rather in coded name-value pairs (“acquisition context”)
- ▶ a mechanism for “annotating” points in the waveform is included in the object; this is achieved by using value types and coordinate references that are similar to structured reporting (but as a list of annotations, rather than a tree)

The different SOP classes are distinguished not only by which features are supported, but also by specific constraints on parameters. For example, the 12-lead ECG SOP class restricts the total number of channels to be no more than thirteen, the number of samples to be less than or equal to 16384, the sampling frequency to be between 200 and 1000 Hertz and the waveform data to be in a signed 16 bit linear form. By comparison, the basic voice audio SOP class supports only one or two channels, specifies no restriction on the number of samples, uses a sampling rate of exactly 8kHz only, and is encoded in 8 bit bytes as either linear, A-law or  $\mu$ -law values.<sup>23</sup>

Mention of the basic audio SOP class in DICOM leads to the next topic, the role of voice in structured reporting.

## **Living with Voice**

In many acquisition applications (such as event logs and measurements), data is inherently digital and already structured. The only issue is how to encode it in a standardized manner for interchange.

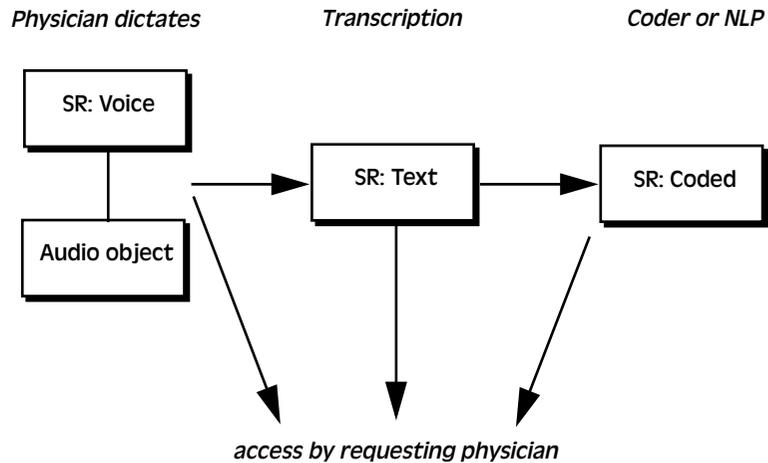
In contrast, for traditional reporting in radiology (as well as surgery, endoscopy, and pathology) the most common practice remains voice dictation. Traditionally, physicians have always generated written material by voice dictation that is subsequently transcribed.

There now exist fairly accurate continuous speech recognition software solutions. Speech recognition can be performed either interactively and directly corrected, or batched in the background, for subsequent review by the author or a “correctionist.” The direct approach has the potential to dramatically reduce the turn-around time for reports, since there is no delay for typing, correction and verification prior to distribution. The idea of integrating diagnostic image review, access to prior images and reports, continuous speech recognition and report generation and verification into a single workstation is extremely attractive. Even better would be a “workflow integrated” solution that supported the use of worklists to drive the review and reporting process.<sup>24</sup>

---

23. A-law and  $\mu$ -law encoding are defined in ITU-T Recommendation G.711. A useful source of information on audio encoding and file formats is Kientzle T. “A Programmer’s Guide to Sound.” Addison-Wesley 1998. ISBN 0-201-41972-6.

24. A DICOM “interpretation worklist” service is currently under development, as is discussed in the chapter on “Document Management: SR in the Real World.”



**FIGURE 29. Dictated Report Workflow**

However, though continuous speech recognition software is increasing in effectiveness, it is most appropriate for use by those who dictate as a high proportion of their work, using standard patterns of speech and language. Diagnostic radiologists fit into this category. Those who only infrequently dictate, such as interventional radiologists, cardiologists and surgeons, may not find continuous speech recognition systems as effective. Nonetheless, continuous speech recognition solutions are rapidly increasing in popularity.

Unfortunately, no matter how it gets turned into plain text, the end result of voice dictation rarely results in a document with any significant structure. Though it is clearly possible to create a DICOM SR that consists entirely of a large blob of plain text, or a single reference to voice audio waveform object, neither approach results in a structured document that is useful for subsequent analysis. Natural language parsing (NLP) algorithms (which extract meaningful structure from plain text) largely remain in the academic realm for now.

DICOM SR can still serve to consolidate both interim and final products in the diagnostic reporting workflow, for example by:

- ▶ encoding the structured inputs to the reporting process, such as lists of images, waveforms and prior reports, logs and automated measurements
- ▶ encoding references to dictated voice audio in waveform objects together with accompanying management information, both as an interim product for distribution and as an input to a transcription process
- ▶ encoding final reports with as much structure as can be gleaned either automatically or manually, including simple section headings, diagnosis codes and so on

It remains difficult with any voice dictation approach to effectively integrate image references, especially when spatial coordinates are required. Even assuming that an

acceptable “pointing” or “drawing” user interface for generating coordinates can be constructed, integrating the captured events in synchronization with the voice stream is not trivial.

A completely different use of speech recognition is for non-continuous speech, specifically voice selection of items from “pick lists” of pre-defined terms and commands. This may have greater applicability in highly structured applications, such as repetitive reporting of a small subset of procedures. Whether it is easier to pick from a list with a pointing device or a spoken word is a matter of personal preference. However, to meet the needs of the many physicians who have difficulty with typing and graphical user interfaces,<sup>25</sup> creative solutions are required.

The role of voice is considered here (in a book on structured reporting), because it is unrealistic to conceive of a practical reporting solution that does not consider it. Whether the priority is to decrease turn-around time or cost by eliminating the transcription or verification process, or to provide access to reports that have been dictated but not yet transcribed, a range of voice solutions need to be supported. Fortunately, the combination of DICOM SR and the basic voice audio waveform SOP class can provide an interoperable mechanism for storing and retrieving reports that are integrated with voice.

## **Living With Paper**

Like it or not, modern information technology has led to increased rather than decreased destruction of forests for the purpose of making paper. The benefits of total conversion to electronic interchange of information are much touted by the enthusiasts. Realistically however, such a conversion will, for most people, be incremental rather than revolutionary, and may never be complete.

Organizations will continue to depend on hardcopy documents for some internal processes, as well as for information received or sent out. In the case of images, film is still commonly used for some applications (such as in the operating room) and modalities (such as mammography). In the case of paper documents, the electronic patient record remains a holy grail rather than a reality for most organizations. Billing and insurance transactions may be closer to a more complete electronic conversion, but paper forms are still filled out by patients in the waiting room. Patients and relatives still receive entire forests worth of printed information in the mail regarding even the most trivial encounter with a health-care provider.

Where does structured reporting fit within this paper-centric world?

---

25. *Or those who are too busy or simply steadfastly refuse to use anything less convenient than a dictaphone. Dictaphones are simple devices that are incredibly easy to use, and few alternatives offer any immediate benefit to the dictating physician.*

## Scanned Documents

DICOM is all about images, yet is remarkably silent on the issue of scanned documents. Outside the closed world of “diagnostic medical imaging,” the term “imaging” to most health-care enterprises and information officers refers to “document imaging.” Specifically, document imaging refers to optically scanning pieces of paper and storing them in electronic form. Document imaging may involve capture of active information, possibly in conjunction with OCR (Optical Character Recognition) and matching of information to pre-defined patterns on forms. Alternatively, document imaging may be used merely as a means of archiving the document before discarding the paper, as an alternative to microfiche for inactive records.

The following characteristics are common to both types of document imaging applications:

- ▶ in a health-care environment, each document is likely to be related in some way to an individual patient for some purpose related to a health-care encounter
- ▶ the document may conform to some well-defined structure, whether it be a form like an admission demographic summary, or a letter to a referring physician
- ▶ the content of the document may need to be referenced during the creation of a subsequent document, e.g. a reference to an operative note from within a diagnostic imaging report

Since these documents are patient related, structured and may be referenced, they potentially fall within the scope of DICOM Structured Reporting.<sup>26</sup>

## Scanned Images

Prior to any other form of processing, a scanned piece of paper exists as a bitmap or continuous tone image (usually grayscale rather than color). Various systems will encode these images in proprietary internal formats or in standard file formats like TIFF (Tagged Image File Format), often with extensions to describe meta-data. The meta-data produced by generic document scanning applications will usually be relatively sparse and confined to page numbers, dates, creators and so on. Health-care specific scanned document solutions may provide mechanisms for inserting patient and transaction related identifying information.

DICOM SR specifically does not have the ability to reference or include by value any such generic image file formats. Before such a scanned image can be referenced directly in a DICOM SR, it must be transcoded into a valid DICOM image.

At the present time, the most appropriate SOP class to use for this purpose is the Secondary Capture object. Before such an object can be created, the image pixel data must be associated with the necessary identifying information for the patient, as well

---

<sup>26</sup> Very likely such documents will exist outside the domain in which DICOM is implemented, and will be encoded as structured documents in some other encoding. The XML-based HL7 CDA (Clinical Document Architecture) is one possibility. Regardless, there will be a need to both generate and access them from the DICOM world.

as some form of study created to hold the information relevant to the episode or transaction to which the document pertains. The Secondary Capture object is a single frame object, so really only one page of a multi-page document can be stored in a DICOM image instance.<sup>27</sup> Once such an image object has been created, it may be referenced by its SOP Instance UID from within a DICOM SR in the usual manner.

The Secondary Capture object is relatively unconstrained in its definition, particularly with respect to the types of pixel data that may be encoded. One can create single bit depth (black and white) images, with the bits efficiently packed 8 per byte or inefficiently packed 1 bit per byte. Grayscale images of any bit depth (e.g. 2, 4, 8, 12 or 16) may be encoded, with zero defined to be black or white. Both true and indexed color images are permitted. In reality, there is very little support amongst medical devices for display of images other than those that are 8 or 16 bit grayscale with a photometric interpretation of MONOCHROME2.<sup>28</sup> Regrettably, there is little if any support in existing devices for single bit images.<sup>29</sup>

What is more, there are no transfer syntaxes in DICOM to efficiently compress single bit scanned document images. The standard lossless JPEG, JPEG-LS and RLE transfer syntaxes will achieve some compression. However, neither the JBIG family of progressive bi-level image compression schemes, nor the ITU T.4 and T.6 facsimile compression schemes (as used by Group 3 and Group 4 fax respectively, and supported in the TIFF Class F profile) are currently defined as standard DICOM transfer syntaxes.

Sufficeth to say, the framework exists to reference scanned document images from within a DICOM SR. Existing DICOM image storage SOP classes are not optimized for this purpose, but may be used. If a real need arises to store and exchange scanned images within a DICOM domain, then more appropriate SOP classes (and transfer syntaxes) can easily be defined.<sup>30</sup> In the interim, an 8 bit grayscale Secondary Capture image is recommended for this purpose.

## Drawings

While on the subject of scanned images, a close relative is the hand drawing. Many radiologists, being visual people, have an artistic bent, and like to illustrate their reports. The spatial coordinate image reference mechanism exists in SR to point to findings on an actual image (the electronic equivalent of the traditional wax pencil).

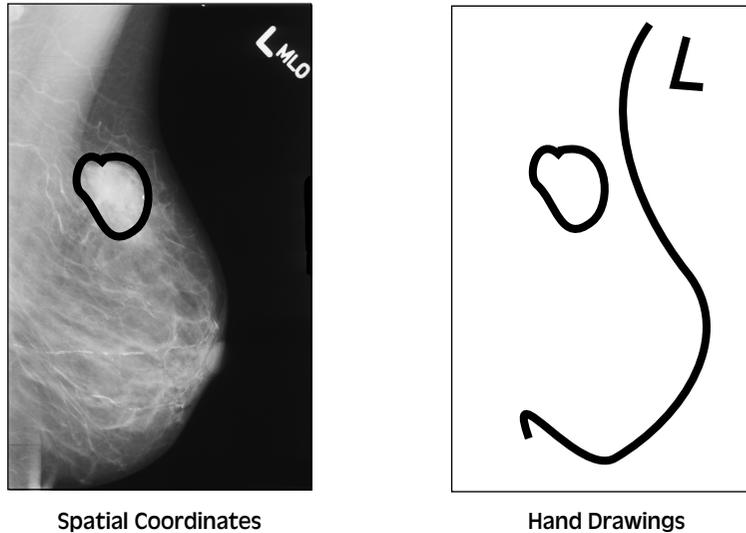
---

27. One could envisage using a simple SR as a collector that referenced multiple individual images and assigned them page numbers, and order, etc.

28. DICOM defines two different grayscale "photometric interpretations" that specify whether the highest pixel value is intended to be rendered as white (MONOCHROME2) or black (MONOCHROME1).

29. Conceivably, a black and white image could be conveyed as a Standalone Overlay object. Overlays can efficiently encode bitmaps, but support for this SOP class is even less common.

30. As of the time of writing, a work item has been approved by the DICOM Committee to revisit the subject of the secondary capture SOP classes. The proposed Supplement 57 defines a family of multi-frame secondary capture objects, distinguished from one another by whether they support single bit, 8-bit grayscale, 12-bit grayscale or color images.



**FIGURE 30. The Role of Drawings**

However, there still exists a need to symbolically convey abstract spatial relationships. For example, it may be more practical to illustrate the location of a lesion relative to adjacent anatomical structures with a crude line drawing, rather than laboriously identifying structures on cross sections or projection images (see Figure 30). A patient's record often contains crude sketches of one thing or another. In some highly formalized reporting applications (such as mammography screening) both paper and electronic forms contain drawing templates intended to be annotated with lesion location.

DICOM SR contains no specific mechanisms for encoding such drawings, either as vector graphics or as bit maps. However, structured reports may reference other objects which contain images of drawings. Indeed, in the section of the standard that describes the various value types available in SR, a note calls attention to the possibility of using either an 8 bit grayscale MONOCHROME2 Secondary Capture image or a Standalone Overlay for this purpose.

In future, it is possible that a standalone vector graphic object might be added to DICOM, perhaps modeled on softcopy grayscale presentation state storage. Possibly, there might also be extensions to the available SR value types in new SR SOP classes.

## Forms

Structured reports are obviously intended for encoding structured information. No document has more structure than a simple paper form of the "fill in the blank" variety familiar to everyone. These forms have proliferated in health-care organizations just as they have in all bureaucracies. Gradually, forms are migrating from printed paper to electronic media, but the structure is similar. DICOM SR is an excellent way

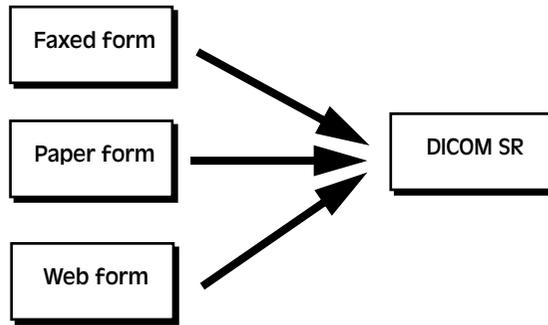


FIGURE 31. Form Recognition and DICOM SR as an End Product

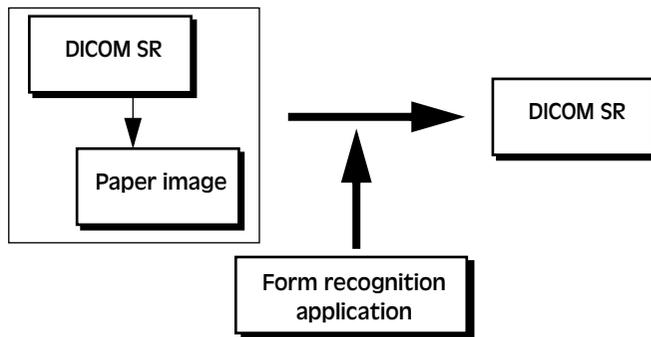


FIGURE 32. Form Recognition and DICOM SR as an Intermediary

of encoding form-based information, given the ease with which categories may be associated with values. Whether the values are free text, chosen from a list, numeric, names, dates and times, or whatever, they can readily be encoded using DICOM SR.

Since optical character recognition is considerably easier when the scanned image conforms to a rigid structure, the task of scanning, interpreting and structuring the content of a completed paper form is well understood. By using structured document formats like DICOM SR as an intermediary, a consistent process for gathering the same information from multiple sources can be devised (see Figure 31).

If a form-based OCR and processing application exists in the DICOM domain, then DICOM SR and image objects can be used in the intermediate steps of the process, much as DICOM SR may be used as an intermediate product with dictated voice stored in DICOM Waveform objects (see Figure 32).

### Acquisition Context

The subject of references to image and waveforms has been considered at great length, both in terms of the mechanics of the references as well as their purpose.

Mechanisms exist for identifying and locating objects by using their unique identifiers. What has not been discussed are mechanisms for *describing* those objects.

The characteristics that describe an image or waveform (in terms of how it was acquired) are grouped together under the heading of “acquisition context.” This is not a term that was used in the DICOM standard until very recently. Traditionally, all attributes that describe an image acquisition are defined as specific data elements. These are listed in the modules from which modality specific information objects are constructed. For example, all of the image objects that involve the use of X-rays contain an attribute called X-ray Tube Current which has a data element tag of (0018,1151). When new information objects are defined, either existing data elements are used when appropriate, or new tags are created. This approach works extremely well when image objects have very specific applications and when the domain from which the descriptive concepts are drawn is well understood and clearly defined. Every standard data element that is added is subject to careful scrutiny during the DICOM balloting and quality control processes.

When the visible light image object family was proposed, an alternative and more general mechanism was introduced. This approach uses lists of coded name-value pairs to specify acquisition content information, rather than depending on SOP-class-specific sets of traditional attributes. Using coded entries has benefits and disadvantages. One of the advantages is the ability to use coding schemes defined by other organizations to define names and value sets for concepts.

For example, most of the recently defined image objects specify the value set for anatomic regions as a context group which contains SNOMED anatomic terms. The information is encoded as a specific DICOM attribute, Anatomic Region Sequence (0008,2218), whose existence pre-dated the introduction of acquisition context. Objects that do not contain this specific attribute could send the same concept as an item of the acquisition context sequence, with a coded concept name that meant “anatomic region.”

This example also illustrates one of the potential risks of allowing an unconstrained set of externally defined concepts and value sets for acquisition context. For interoperability to be achieved, DICOM has relied heavily on attributes with well-defined meanings, constrained value sets and specific requirements on their presence in particular objects. A generic acquisition context mechanism reduces the control that DICOM can exert. It introduces the possibility that different implementations may use different codes for concepts and values and not be able to communicate. For acquisition context that is purely “annotative” and only intended to be read by a human, this is not a significant problem. This is not the case when the context is used as input for an automated decision, such as where to send an image or which way around to display it. Such considerations are taken into account by designers of new objects when determining whether to introduce or reuse specific data elements, or to use the generic acquisition context mechanism.<sup>31</sup> These are not the only alternatives, since one can also define constrained sets (enumerated values) for both concept names and values of acquisition context items.

Acquisition content in an image or waveform is encoded as a list of sequence items in an Acquisition Context Sequence (0040,0555). This is a flat list and has no tree structure, though each sequence item is very similar to an SR content item.<sup>32</sup> In particular, each item consists of a name-value pair. The name is specified an single coded entry item of Concept Name Code Sequence (0040,A043), just as in SR. The corresponding value may be one of a range of value types, though no specific value type attribute is used. Rather, the type of value is indicated by which value attribute is present in the acquisition context sequence item. The range of attributes used to convey the value include numeric values with units, dates, times, person names, text values and coded entries. The same data elements that are used to encode SR values are also used in the acquisition context. A free text description of the context item may be provided. A set of frame numbers can be included to constrain to which frames the context item applies; by default it applies to the entire object.

As an example, consider a digital chest X-ray image which contains acquisition context information beyond that encoded in the traditional DICOM objects, such as whether or not the image was obtained during inspiration or expiration:

```
(0040,a555) Acquisition Context Sequence
(fffe,e000) Item
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209702"
(0008,0102) Coding Scheme Designator "99PMP"
(0008,0104) Code Meaning "Functional condition"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "F-20010"
(0008,0102) Coding Scheme Designator "SNM3"
(0008,0104) Code Meaning "Inspiration"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
```

---

31. The digital X-ray objects take a middle ground. They use many specific DICOM attributes (such as those that describe technique, positioning and equipment) but encode more "subtle" information using acquisition context (such as flexion or extension of the body part). This is partly a consequence of their derivation from older objects with the same data elements. The visible light and waveform objects, being entirely new as well as less specific to a single clinical domain or technique of acquisition, rely more heavily on acquisition context.

32. The similarity is not incidental. Both structured reporting and acquisition context (as originally proposed for visible light (VL) images) were largely the work of Dean Bidgood, assisted by the members of the SR and VL working groups, many of whom were from outside the "traditional" DICOM community.

This information is quite distinct from information in the same image that is required to be encoded using traditional DICOM data elements. There are a great many of these, including those already mentioned:

```
...
(0018,1151) X-ray Tube Current "70"
...
(0008,2218) Anatomic Region Sequence
(fffe,e000) Item
(0008,0100) Code Value "T-D3000"
(0008,0102) Coding Scheme Designator "SNM3"
(0008,0104) Code Meaning "Chest"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
...
```

The rules for acquisition context specify that the acquisition context sequence shall not contain descriptions of conditions that replace those that are already described in specific attributes that are also contained within the IOD. This is to avoid the potential for duplication, ambiguity or conflict.

## Acquisition Context in SR

The preceding section described how acquisition context may be encoded in an image or waveform object. Different entirely is the encoding of acquisition context about images and waveforms that are referenced from within a structured report.<sup>33</sup>

Suppose for example, that a mammography image is referenced from within a mammography report. An application that is analyzing or rendering the report may need to know what radiographic projection was used to obtain that particular image. Ideally, the application should not have to retrieve the image and parse it to determine the projection. Indeed it may not be capable of doing so, nor even have access to the image. The alternative is to copy any relevant information about how a referenced object was acquired into the structured report. This information is distinguished from other forms of content by the use of a specific relationship, "HAS ACQ CONTEXT". For example, the information that a mammography image is a medio-lateral oblique (MLO) projection can be extracted from the image and encoded in the report as a child of the image reference itself:

```
<IMAGE:=(MG,"1.2.3.4")>
  <has acq context CODE:(,, "View")=(,, "MLO")>
```

This sounds simple, but in practice two issues arise:

- ▶ much of the information in the images is not in the form of coded entries but traditional DICOM data elements and values

---

33. Notice that a structured report itself has no "acquisition context" per se. Any information that is arguably in the same category as acquisition context that describes the generation of the report itself is encoded either in the SR Document General module, or as observation context.

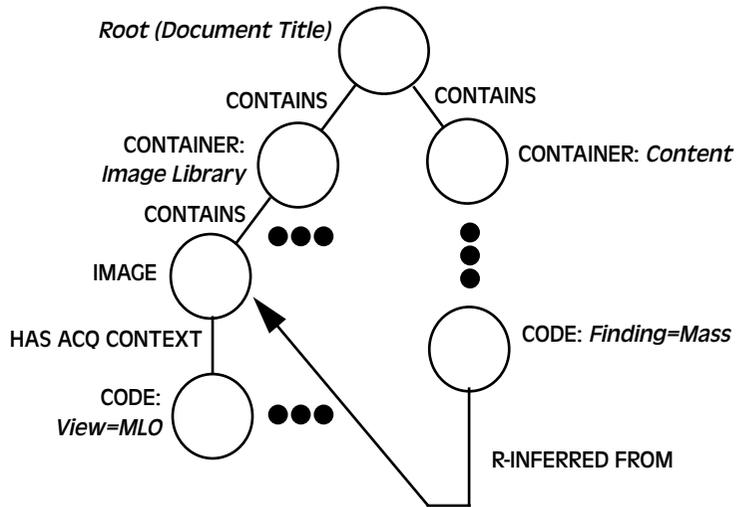


FIGURE 33. “Library” of Image Acquisition Context

- reports that contain multiple references to the same objects may unnecessarily duplicate acquisition context information in each reference

An elegant solution to the second problem is to define a sub-tree of the root node that is a “library” of image and waveform references with their associated acquisition context, as illustrated in Figure 33.<sup>34</sup> “By-reference” relationships can then be used to point into the library when references occur in the “content” sub-tree of the root node. With this approach, it is not always easy to decide what to use for a purpose of reference concept name and where to encode it. Nor is it always obvious what reference relationship type to use, “has properties” or “inferred from” or “contains.”

This method also only works for those SOP classes that permit by-reference relationships. Otherwise, duplication and redundancy is difficult to avoid. Astute readers will have noticed that the use of an object library approach leads to three different places where image and waveform object references may occur: in the library, where the reference occurs in the content, and in the management information in the SR Document General Module.

The other problem with the acquisition context is how to transform traditional DICOM data elements and values into SR content items with coded concept names and SR value types. The issue is not the mechanism by which the name-value pairs are encoded, since this is obvious. Rather the problem is where to find a set of codes that corresponds to the traditional DICOM data elements. There has been a general reluctance

34. This suggestion was made by DICOM WG 15 Digital Mammography and arose during the course of development of the Computer Assisted Detection (CAD) SR object. This object involves many image references, potentially hundreds of them. See Supplement 50, public comment draft dated June 28, 2000.

tance to simply map elements of the form “(gggg, eeee) meaning” into coded entry tuples of the form “(gggeee, DICOM, meaning).”<sup>35</sup> Apart from aesthetic difficulties, such a general “construction” approach to generating codes runs the risk of overlapping with existing codes for concepts in other coding schemes. One of the goals of switching to coded entries in the first place is to harmonize with other standards, and gratuitously creating an equivalent concept for every DICOM data element is not in keeping with this goal.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ the “text+image” approach to reporting (to which unenhanced web reporting applications are limited) can be implemented with DICOM SR, but the additional feature of coordinates makes structured reports more useful
- ▶ the use of coordinates is analogous to the “wax-pencil” paradigm of the film world, but goes one step further by associating specific locations with specific observations
- ▶ image references in structured reports may also contain presentation states that convey how an image should be rendered (in terms of contrast and spatial transformations) in order to achieve consistent appearance on different devices
- ▶ the “selected from” relationship is used to associate either temporal or spatial coordinates (or both) with an image or waveform
- ▶ DICOM defines objects for electrocardiographic and hemodynamic waveforms, as well as voice audio
- ▶ the voice audio waveform objects can be used with SR objects to integrate dictated text into the DICOM domain
- ▶ despite advances in speech recognition and a desire to structure content, the pervasiveness of plain text dictation cannot be ignored
- ▶ there is a significant role for scanned paper documents such as old medical records, forms and drawings; though DICOM does not directly address document imaging, secondary capture objects can be used if necessary
- ▶ acquisition context is information about how a referenced image or waveform was acquired
- ▶ images and waveforms include acquisition context in the form of traditional DICOM data elements as well as by a sequence of coded name-value pairs
- ▶ reports may include, by value, acquisition context about referenced objects, using the “has acquisition context” relationship

---

35. This very pragmatic suggestion came from Harry Solomon during the course of his work on the waveform supplement in WG 1. It has been decided not to adopt this approach, but rather to define new (arbitrarily numbered) codes for existing traditional attributes as they are needed.

# ***SOP Classes and Conformance***

---

One feature that distinguishes DICOM from most other medical informatics standards is an obsession with conformance. The “unit” of conformance in DICOM is the Service-Object Pair (SOP) Class, which specifies an Information Object Definition (IOD) paired with a Service. In the case of composite IODs such as images and structured reports, the service in question is the Storage Service Class.<sup>1</sup>

## **IODs, SOP Classes and Conformance**

The choice of IOD distinguishes the different Storage SOP classes. Thus there is a CT Image Storage SOP class for Computed Tomography images, an MR Image Storage SOP class for Magnetic Resonance images, and so on. Each has a corresponding IOD paired with the single common Storage Service Class. All these composite IODs share information describing:

- ▶ the patient, such as name, date of birth, etc.
- ▶ the study, such as ID, UID, etc.
- ▶ the series, such as number, UID, etc.
- ▶ the individual instance, such as number, UID, etc.

Where the specific IODs differ is in the constraints on, and the description of, information related to the specific “modality.” For example, a CT Image IOD will contain an instance level attribute like Exposure Time, since it is an X-ray based modality. An MR Image IOD will contain an instance level attribute like Repetition Time to describe a parameter of an MR pulse sequence. There are usually also constraints on

---

1. *There is also a Media Storage Class that essentially mirrors the network Storage Service Class for the purpose of storing objects on removable interchange media.,*

the form of pixel data specified in each IOD. A CT image is limited to 12 to 16 bit grayscale, for example.

The intent of these constraints is to make sure that senders of images (storage Service Class Users (SCUs)) always create objects in a consistent way, meeting mandatory requirements, to avoid gratuitous differences that reduce interoperability. Conversely, receivers of images (storage Service Class Providers (SCPs)) can always expect to find the same information encoded in the same form in the same attributes, based on the SOP class (and corresponding IOD).

The SOP classes supported by a device are required to be specified in its Conformance Statement. The SOP classes supported are also negotiated over the network when an Association<sup>2</sup> is established between two devices. In the case of interchange media, the SOP classes supported by a particular Media Application Profile are specified a priori.<sup>3</sup> This SOP class based conformance mechanism is one of the keys to DICOM's success as a standard, and distinguishes it from the earlier ACR-NEMA standards which lacked such a mechanism. These earlier standards specified a dictionary from which attributes could be chosen, and a generic encoding mechanism, without further constraining content on a modality specific basis.

## SR IODs and SOP Classes

The image storage SOP classes so far described are naturally divided on the basis of modality. Not all storage SOP classes are for images, however. There are SOP classes for stand-alone objects like Curves and Overlays. Though not widely used, these established the precedent for using the same composite patient/study/series/instance information model to store things other than pixel data. There is now the Grayscale Softcopy Presentation State Storage SOP class for storing information to be applied to an image for presentation (such as contrast and geometric transformations). An entire family of time-based waveform storage SOP classes has recently been added. These provide support for voice audio, various forms of electrocardiography and cardiovascular hemodynamic waveforms.

Choosing SOP classes for Structured Reporting presented a slightly different challenge. To start with, the range of applications of SR is vast. Furthermore, the framework that SR introduces is both novel and complex. Three different approaches were considered. One approach would have been to follow the image and waveform model, and introduce application specific SOP classes as the need arose. This would have maximized interoperability by constraining the form and content of the SR tree, but would have led to a proliferation of SOP classes, most of which would have a lot in common, and most of which could be created or rendered by the same applications. A second approach would have been to have a single completely generic SOP

- 
2. The DICOM word for a connection between two devices. Strictly speaking it is the ISO OSI term for a connection (actually a "cooperative relationship") between two "Application Entities" (AEs).
  3. The exception to this is the General Purpose CD-R profile, which allows any Media Storage SOP class to be used, with the intent that supported SOP classes be specified in the Conformance Statement.

class that allowed the full power of SR (including relationships by reference, numeric values and spatial coordinates) to be used in every case. This approach would have made life very easy for the creators of reports, but very hard for the consumers of reports, which would have had to be prepared to handle every SR feature.<sup>4</sup> This was considered to be far too much of a burden for many simple applications that really require very little of the power of the full SR framework.

The approach which was finally adopted is a compromise that defines three general purpose SR SOP classes of increasing complexity. The intent is that these be used for most applications, or serve as the basis of application specific SOP classes that use templates to further constrain content rather than features.<sup>5</sup>

**TABLE 5. Structured Reporting IODs and SOP Classes**

Basic Text Structured Report
Enhanced Structured Report
Comprehensive Structured Report

Each of these SOP classes is a proper subset or superset of the other. Thus one could re-encode a Basic Text SR instance as a valid Enhanced or Comprehensive SR instance. Similarly, an Enhanced SR instance is a valid Comprehensive SR instance.

The following features distinguish these SOP classes:

- ▶ whether or not relationships by reference are permitted
- ▶ whether or not numeric measurements may be used
- ▶ whether or not spatial or temporal coordinates may be used
- ▶ whether or not references to images and waveforms are restricted to leaves
- ▶ whether non-leaf content is restricted to containers and text only, or whether codes may be used as the source of relationships like “has properties” and “inferred from”

4. In the case of structured reports, every SCP is required by the standard to be able to “convey its full meaning in an unambiguous manner,” as will be discussed later.

5. It is expected that some applications with very specific interoperability requirements will define completely new SOP classes, with no more or less features than are required for a very specific task. An example is the proposed Mammography Computer Assisted Detection SOP class which will use the SR framework, but in a highly constrained manner. This is necessary because the application is very precisely defined and subject to very stringent safety and regulatory requirements.

TABLE 6. Summary of SR SOP Class Specific Constraints

Feature	Basic Text	Enhanced	Comprehensive
by reference relationships	-	-	Y
numeric measurements	-	Y	Y
spatial & temporal coordinates	-	Y	Y
non-leaf image & waveform references	-	-	Y
non-leaf more than just text & containers	-	-	Y

All of the SOP classes provide the following core functionality however:

- ▶ concept names (used for the document title, headings, purpose of reference and names of name-value pairs) which are always coded entries, not free text
- ▶ value types of container, text and code
- ▶ value types for persons names, dates, times and UID references
- ▶ concept modifiers using codes or text (though the complexity of the modifier is constrained by the other restrictions on code and text relationships)
- ▶ observation context and acquisition context relationships
- ▶ the capability to reference images and waveforms

The constraints for each SOP class and corresponding IOD are documented using a new convention. In the past, IODs have been defined by using modality or application specific “modules” that describe which attributes are to be used and what their value sets are. For SR, a completely generic set of modules are used, which provide the tools from which to build new SR IODs. The IOD definitions themselves contain tables that restrict:

- ▶ the range of allowed value types
- ▶ the range of allowed relationship types, for specific source and target value types

For example, in the Enhanced IOD, the use of the “selected from” relationship that links coordinates with their composite objects is specified separately for spatial and temporal coordinates. In particular, spatial coordinates may only be selected from images. Temporal coordinates may be selected from spatial coordinates, images and waveforms. Thus if one wants to specify spatial and temporal coordinates selected from an image, the temporal coordinates must be specified first (highest in the tree), then selected from the spatial coordinates, and not the other way around.<sup>6</sup>

## What does conformance actually mean?

In the case of image storage SOP classes, conformance has mostly been an issue for SCUs, which have to make sure they send information in the prescribed form. There

6. Constraints like these are present not to restrict flexibility, but to avoid gratuitous incompatibility. If two ways to do the same thing are possible, one way will be forbidden, whenever feasible.

has been little emphasis on SCP conformance, since there are no display requirements specified in the standard. It isn't very difficult to store a DICOM image, which is about all that the standard specifies. In practice however, users expect that an SCP will "do the right thing" with an image of a particular SOP class. From the perspective of a workstation or PACS, the expectation, even if it isn't fully elucidated in the standard, is to be able to display any image complying with the IOD. So for example, it is not acceptable to a customer to fail to display images with extreme or unusual pixel values, such as a 16 bit unsigned MR image that really does use all 16 bits, just because one's internal representation treats 16 bit values as signed.

Increasingly, as new SOP classes are defined more rigorous conformance requirements are specified for the SCP, "raising the bar," so as to speak. This is partly because of lessons learned from past mistakes, but also because users now have higher expectations. It is easier to reach consensus when vendors are under greater pressure from customers to meet standard connectivity requirements.

For structured reports, conformance as an SCU is straightforward. The limits on how one may use value types and relationships are defined for a particular SOP class. This still leaves enormous flexibility in terms of which codes to use to convey concepts and how to structure the tree.

This flexibility creates quite a burden for the SCP. The standard states that an SCP is required to be a Level 2 SCP.<sup>7</sup> This means that the SCP cannot throw any part of the tree away. The standard also states that an SCP that intends to "display or otherwise render" a report (as opposed to merely archiving and regurgitating it) is required to "convey its full meaning in an unambiguous manner." No further definition of this requirement is stated, but it is pretty clear from this statement that an SCP:

- ▶ cannot ignore parts of the tree that it doesn't like or are nested too deeply
- ▶ cannot ignore value types that it doesn't like
- ▶ cannot ignore concept modifiers
- ▶ must fully elucidate both the concept name and the value of name value pairs, since displaying the value alone may not convey the meaning
- ▶ cannot ignore relationship types, since the type of relationship alters the readers perception of the meaning of the nested content
- ▶ must be wary of how observation context is propagated via inheritance to descendants in the tree

Some examples may help to clarify some of these issues.

---

7. One may specify in the conformance statement and negotiate during association establishment the "level" of storage by an SCP. This allows the SCU to know whether the SCP will always preserve attributes that are optional or mandatory. For SR SOP classes, only Level 2 is permitted. This means that all optional and mandatory standard attributes must be preserved. Only private elements may be discarded.

## Conformance and Negation Related Issues

**Not yet  
standardized.  
Final standard  
may differ  
substantially.**

The concept of negation is crucial in many applications. It is equally important to be able to say “no metastases” as it is to say “metastases.”<sup>8</sup> The potential alternatives for describing negation illustrate some of the pitfalls inherent in rendering a structured report “unambiguously.” At the present time, there is no standard tool for encoding negation, and so one must fall back on the use of specific coded concept names or values, or use of the concept modifier relationship.<sup>9</sup>

### Negation using Pre-coordinated Concept Names

Consider a report that contains the following items:

```
<CONTAINER:“Findings”>
  <contains TEXT:“Finding”=“enlarged heart”>
  <contains TEXT:“Finding”=“widened mediastinum”>
```

It might be tempting to render this as a bulleted list, eliding the apparently redundant “finding” concept name, as:

```
Findings:
- enlarged heart
- widened mediastinum
```

or, if one is clever enough, to translate this into natural language:

```
Findings: There is an enlarged heart and a widened
mediastinum.
```

However consider a similar pattern:

```
<CONTAINER:“Findings”>
  <contains TEXT:“Not found”=“enlarged heart”>
  <contains TEXT:“Not found”=“widened mediastinum”>
```

which should be rendered as:

```
Findings:
- no enlarged heart
- no widened mediastinum
```

or:

```
Findings: There is no enlarged heart and no widened
mediastinum.
```

8. Radiologists are also particularly fond of listing things that they do not find, even on entirely normal examinations. The report then reads as though some work was actually done, and value has been added. Ridiculous as it may sound, it is often the perception of referring physicians that reports that just say “normal” are somehow inferior or incomplete. To be fair, particularly for trainees, the discipline of enumerating potential findings that are not present may be helpful.

9. The subject of negation is revisited in the chapter on “Templates: Bringing order to chaos ...”. It is hoped that in future the standard will define standard templates for negation that will reduce ambiguity and allow consistent encoding and rendering of negated concepts.

or similar.<sup>10</sup> However, if an application blithely assumed that everything in a container with a concept name of “Findings” was a positive finding, then it might be rendered incorrectly as:

```
Findings:
- enlarged heart
- widened mediastinum
```

In such a rendering, the negation concept has been lost.

To put this another way, the safest thing to do is to *always explicitly render the concept name*. If the rendering application can specifically recognize the concept name codes that are used to indicate individual items, and match them with redundant parent concept names, only then can it do better.<sup>11</sup>

Specifically, if one doesn’t recognize the code for an individual “Finding” as being an item of a container of “Findings” then one should go ahead and render the full name-value pair. For example:

```
Findings:
- Finding = enlarged heart
- Finding = widened mediastinum
```

This may be untidy, but it is always safe and unambiguous.<sup>12</sup>

## Negation using Concept Modifiers

So far, pre-coordinated codes have been used to convey the concept of negation. Specifically a single code that means “not something” rather than “something” has been used.

An alternative approach is to modify the concept of “something” with a modifier of “not”. Clearly this gives rise to similar rendering issues as the first approach, which is why the matter is raised in the context of SCP conformance and behavior.

As in the previous example, to convey the notion that an individual finding is not present rather than present, one could write:

```
<CONTAINER:“Findings”>
  <contains TEXT:“Finding”=“enlarged heart”>
    <has concept mod CODE:“Presence”=“Absent”>
  <contains TEXT:“Finding”=“widened mediastinum”>
    <has concept mod CODE:“Presence”=“Absent”>
```

10. This is not meant to imply that there is (yet) any such code as “Not found” in any standard dictionary for concept names. Neither does this mean that one could not define such a code, for example, in a private dictionary.

11. Though an SCP is required to rendered every report unambiguously, if it recognizes certain patterns or templates, it may be able to render some reports more “beautifully” than others.

12. Obviously the use of SR will greatly benefit from the adoption of standard codes and templates for such common patterns, allowing general purpose display applications to do better more often.

The worst-case rendering should look something like:

Findings:

- Finding (Presence=Absent) = enlarged heart
- Finding (Presence=Absent) = widened mediastinum

Although this is untidy, it is safe and unambiguous. An application that can recognize the “meaning” behind the codes used here could do much better and render something like:

Findings:

- no enlarged heart
- no widened mediastinum

In the context of conformance, the point is simply that SR SCPs *must never ignore concept modifiers*.

## Observation Context

SCPs must also apply the rules for inheritance of observation context, if they wish to be able to call out some attribute of observation context at a nesting level deeper than where the attribute was explicitly defined.

For example, if an attribute of observation context like “Observer” is defined as a child of the root node, it is inherited by the content of the entire SR document tree. At some nested inner node, if the rendering application wishes to display the “Observer” applicable to that node, it must be capable of finding where it is defined in accordance with the inheritance rules.<sup>13</sup>

The subject of observation context is discussed in more detail in the appropriate chapter.

## Conformance and References to Other SOP Instances

SR documents may reference external objects, such as images, waveforms and presentation states. A device or application is required to specify in its conformance statement which SOP classes are supported for such references. When an association is established, an SCU can determine which storage SOP classes are supported by the SCP, and theoretically restrict references from within an SR document accordingly.<sup>14</sup>

Nevertheless, an SCP has to be prepared to render a report that contains references to types of objects that it does not understand and is not able to display. For example, an image report display workstation is very likely not going to be able to display a waveform that has been referenced in a structured report.<sup>15</sup> How an application han-

---

13. Clearly this is more of an issue when observation context “diverges” deeper in the tree. The classic example is the use of one subtree for fetus A and another subtree for fetus B in an obstetric ultrasound report on twins.

14. In reality, creation of an SR object and its subsequent transmission are likely to be performed by different applications, processes or layers of the software. Thus the choice of what types of object to reference will likely be determined a priori by configuration, rather than dynamically.

dles this situation must be documented in the conformance statement. How the application will render such a report “unambiguously” is undefined. Not rendering it at all is probably not an acceptable option to the user.

## Other Conformance Issues

Another aspect of conformance and SCP behavior that is defined specifically for Structured Reporting, is when to create a new SOP Instance and assign a new SOP Instance UID. This will be discussed in the chapter on “Document Management: SR in the Real World.”

## Summary

Some of the key points mentioned in this chapter include:

- ▶ the SOP class is the “unit” of conformance in DICOM when paired with the role of user (SCU) or provider (SCP)
- ▶ from the perspective of images, waveforms and structured reports, a specific IOD is paired with the generic Storage Service Class to create specific SOP classes
- ▶ structured reports are stored using one of three SOP classes of increasing complexity: Basic Text SR, Enhanced SR and Comprehensive SR
- ▶ the Basic Text SR Storage SOP class supports containers, text, code and image and waveform references, concept modifiers, observation and acquisition context, and contains relationships (primarily for headings)
- ▶ the Enhanced SR Storage SOP class adds numeric measurements and coordinates
- ▶ the Comprehensive SR Storage SOP class adds by-reference relationships and allows most value types to occur within the tree rather than only as leaves
- ▶ conformance as an SCP for SR objects means more than just storing the object; there is also a requirement for unambiguous rendering
- ▶ SR storage SCP’s must not discard optional attributes (i.e. they are always level 2)
- ▶ rendering applications may not ignore concept modifiers, since modifiers may completely change the meaning of a report (e.g. by negating a concept)
- ▶ inheritance of observation context may need to be considered by a rendering application
- ▶ an application may not be capable of displaying all objects (such as images and waveforms) that are referenced by a report, and must be capable of handling this situation gracefully

---

15. *Even if an application can render all currently defined storage SOP classes, new SOP classes will certainly be defined in future versions of the standard, and private SOP classes also exist. Thus every application has to be able to handle unknown referenced SOP classes gracefully.*



# ***Context: Who did what to whom, when...***

---

A document consists of more than just content. It is also important to know who or what created that content, and in what “context.” This may involve describing:

- ▶ who (or what)
- ▶ did what
- ▶ to whom (or what)

or to put it in more technical terms, identifying the:

- ▶ observer context
- ▶ procedure context
- ▶ subject context

In this chapter, these aspects of observation context will be discussed.<sup>1</sup> Where the context is defined, how it is encoded, how it defaults, and how it is inherited in a top-down traversal of the SR tree will be addressed. The concept of different “modes” of observation will be defined.

Observation context is specific to structured reporting and distinct from “acquisition context,” which applies to images and waveforms. Acquisition context may be included within in an SR document to encode information present in a traditional



---

1. *What is described here is based on the SR supplement itself, which is now part of the standard, as well as proposals for templates for observation context that have not yet been finalized. In addition, where issues of inheritance are vague or currently undefined, concepts are drawn from earlier drafts, documents and articles by Dean Bidgood. Be warned that this area of the standard is in a state of flux. In particular, the rules regarding inheritance of context may change. The specific attributes of observation context may not precisely correspond to what is presented here. The rules regarding quoted context may differ in any final standard.*

DICOM “acquired” object such as an image or waveform. This subject is covered separately in the chapter on “Images, Waveforms, and Voice Audio.”

## Observer Context

The “observer context” identifies and describes “who or what is making an observation.” Perhaps the simplest case is one in which a physician is creating a report that consists entirely of observations and inferences they themselves have made. In such a case, little more need be stated than the physician’s name and possibly to what organization they belong or are responsible. The purpose is to be able to state unambiguously who generated the content. In the terminology of DICOM SR, such a person is referred to as the “observer.”

Note that the “observer” of the content is not necessarily the same person as the “verifying observer.” Indeed a common scenario is for a junior physician to create a report, and a more senior one to verify it (and hence take responsibility for it or attest to the veracity of its content). The definition of the verifying observer<sup>2</sup> is not part of observer context; rather it is document management information, and as such is stored elsewhere outside the SR tree.<sup>3</sup>

The observer is not necessarily a person. In many cases it may be a device or some software. For example, an ECG device that knows how to identify the locations of the P waves in a waveform may include such observations in an SR document that is its output. In this case, the observer would be identified as the device, not the person operating the device. It is also conceivable that more than one observer may contribute to content, or that there may be both a person and a device making an observation.

In summary then, observer context consists of information about:

- ▶ the person who made the observation
- ▶ the device that made the observation
- ▶ the organization to which they belong

An important question arises if a more complex SR document tree consists of items created by different observers. This requires discussion of inheritance of observation context, which will be addressed later in this chapter.

Also covered later is the question of how one encodes context in the SR document tree, and where the definitions of the context are standardized.

---

2. Or observers... there may be multiple verifying observers, for example in a case that is “double read,” such as a screening mammogram.

3. Specifically, the verifying observers are defined in the SR Document General Module as attributes of the Verifying Observer Sequence (0040, A073).

## Subject Context

The “subject context” identifies and describes “on whom or what the observation is made.” In the simple case of a conventional adult radiology report, the “subject” is the patient who underwent the procedure that is being reported. Even if results of previous procedures are being correlated with the current procedure, the observation subject is still the same patient.

More complex situations arise when the observations being made are not clearly about an entire single individual human patient. Examples include:

- ▶ one or more fetuses contained within a mother
- ▶ a surgical specimen removed from a patient
- ▶ a forensic specimen, possibly not (yet) associated with an actual person

The approach used in DICOM SR is to keep things simple for the most common cases, but to include the flexibility to precisely define the more complex cases as necessary. Thus, as will be described in a later section describing initial (default) context, when the subject is the same as the patient named in the composite object General Patient Module, nothing more need be said.<sup>4</sup> When it is necessary to define the subject more precisely, attributes of subject context may be encoded in the SR document tree.

For example, consider the case of the obstetric ultrasound report already mentioned. It is necessary to make observations about each fetus independently. Such observations include measurements of the dimensions of body parts that are used to estimate gestational age, observations about normal or abnormal anatomical findings and so on. One therefore needs to be able to identify twin 1 of 2, and twin 2 of 2. Ideally, when comparing an earlier examination with the current examination, one would like to be able to refer to the same fetus as before, for example to say that one is developing normally and the other is not. Unfortunately since fetuses don't come conveniently labelled with ultrasonically visible bar codes, this is not always as easy as it sounds. Thus the subject context attributes that identify a fetus need to be able to distinguish between a particular fetus that has been reliably identified, and one that has not, and has been arbitrarily assigned an identifier on each occasion.

The specific attributes of subject context depend on what the subject actually is. For example, a whole patient is identified in a different way than a fetus or a specimen is. Thus one of the key attributes of subject context is the “subject class.” Other attributes of subject context are predicated on this class being one of a number of predefined coded values, which include:

- ▶ patient
- ▶ specimen
- ▶ fetus

---

4. The same is true when the subject context for a specimen is sufficiently defined by the Specimen Identification Module.

If the subject is a patient, then the usual repertoire of patient identifying attributes that mirror those in the General Patient and Patient Study Modules may be used. These include name, id, date of birth and so on. As has been mentioned, fetuses can be assigned numbers or names related to their multiplicity as necessary. Specimens may be identified using attributes with coded concept names that mirror the attributes of the Specimen Identification Module.

## Procedure Context

The “procedure context” identifies and describes “what was done.” More precisely, it provides a means of specifying:

- ▶ the “data acquisition procedure” that this document is about
- ▶ the “interpretation procedure”

The most common case is a single report that is about a single diagnostic procedure. In this case, the procedure context defaults to the images, waveforms and other composite objects that are part of the same study as identified in the General Study Module. For example, an SR Document that is a report on a chest X-ray, would normally share the same Study ID, Accession Number and Study Instance UID as those in the images; the procedure context is obvious and unambiguous and nothing more need be said.

However, if a report contains observations about more than one identifiable procedure, or summarizes information about multiple procedures, then it is necessary to unambiguously define which procedure a particular observation is about. The proposed templates that define procedure context introduce the notion of the “current” data acquisition procedure, which is the one with the same study identifier as the report, and “additional” or “other” procedures.

A distinction is made between “data acquisition procedure” and “interpretation procedure.” These may not always be synonymous. In simple situations, the identifiers will be the same. This will be the case when the SR object contains information (such as measurements) derived from images as part of the acquisition procedure, or when the report is actually an interpretation of the “current procedure.” Reports that are created in response to different requests, such as to review a previously reported study, will have different identifiers. Specifically, the attributes of the General Study Module will only identify the current interpretation procedure. Content from the studies being reviewed that is referenced or quoted will be identified with attributes of procedure context.

## Identification vs. Description

It should be apparent from the foregoing sections that the focus of observation context is on identification rather than description. It is important to identify “who made an observation” unambiguously, but perhaps not important to describe their intimate personal details in every report. Less obviously, the procedure context serves to identify the data acquisition and interpretation procedures, rather than describe what was done. For example, the attributes of procedure context include the identifiers of a

particular X-ray examination, but not a description of what kind of examination was performed. Such information is deemed to be more properly the province of acquisition rather than observation context. Acquisition context is described elsewhere (in the chapter on “Images, Waveforms, and Voice Audio”).

## Encoding Observation Context

The categories of observation context have been described in general terms. Much of it defaults to values that are already encoded in top level attributes just like any other DICOM Composite IOD. When the observation subject is the entire patient, then the Patient’s Name (0010, 0010) and Patient’s ID (0010, 0020) and any other attributes of the General Patient Module serve as identifiers in the same way as they do as for an image or waveform object. For example:

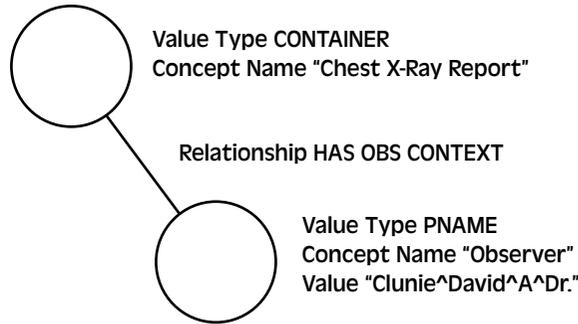
```
...
(0010,0010) Patient’s Name VR=<PN> “Smith^Mary”
(0010,0020) Patient’s ID   VR=<SH> “99-2562-197”
...
```

For observation context that is encoded in the SR tree, the normal content encoding mechanisms are used. This is a key aspect of observation context: it is just the same as any other content, with the one exception that there are inheritance rules about its top-down propagation and its defaults. How then does one distinguish observation context from other content? By use of a specific “HAS OBS CONTEXT” relationship.<sup>5</sup>

The “HAS OBS CONTEXT” relationship is defined to modify the observation context of the parent node. That is the accumulated observation context of the source node of the relationship (and all its other descendants) is modified by the content defined in the child (target) node. For example, if the parent node is the root node of the whole document, then any children with a “HAS OBS CONTEXT” relationship define the observation context for the whole document. This is not a special case per se, just a consequence of the definition of the behavior of the relationship. It is a pattern that applies often.<sup>6</sup> The example illustrated in Figure 34 is encoded as:

```
<CONTAINER:(209076,99PMP,“Chest X-ray Report”)>
  <has obs context PNAME:(209069,99PMP,“Observer”)
    =“Clunie^David^A^Dr.”>
```

- 
5. One might well ask why the relationship type is abbreviated to “HAS OBS CONTEXT” rather than use “HAS OBSERVATION CONTEXT”. The value representation for the Relationship Type (0040, A010) attribute is Code String (CS), which is limited to 16 characters.
  6. For simplicity’s sake, the requirements for structured reporting for the Integrating the Healthcare Enterprise (IHE) demonstration in 2000 specify that “has observation context” relationships may only be used with the root node as the source.



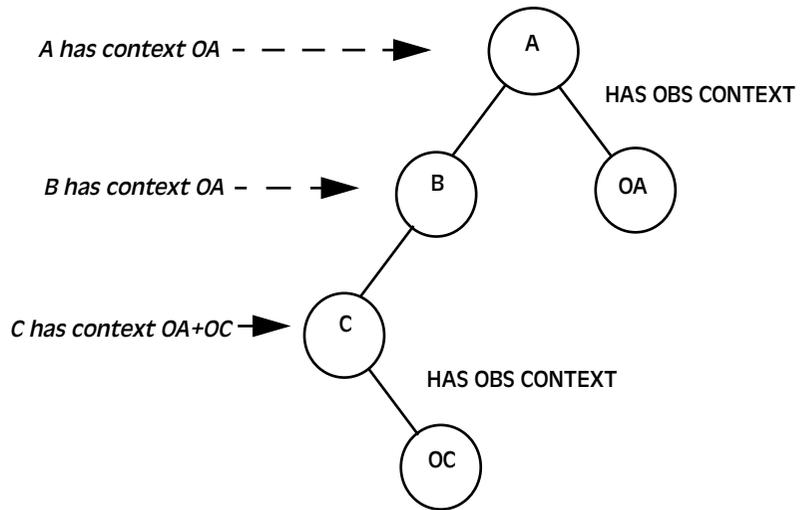
**FIGURE 34. Example of "Has Observation Context" Relationship**

---

The child (target) node is specifying an attribute of observation context for the parent (source) node. As will be discussed, this attribute will be inherited by all the other children of the parent node. The attribute of observation context has been encoded in the same manner as any other content. Specifically, it is a name-value pair, the concept name is a coded entry (in this case "Observer"), one of the value types for this concept specified in the template for observation context is PNAME, and the name of the observer has been specified just as for any other person name value.

For completeness, a more detailed dump of how this content item would be encoded follows:

```
(0040,a040) Value Type "CONTAINER"  
(0040,a043) Concept Name Code Sequence  
(ffff,e000) Item  
(0008,0100) Code Value "209076"  
(0008,0102) Coding Scheme Designator "99PMP"  
(0008,0104) Code Meaning "Chest X-ray Report"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0040,a050) Continuity of Content "SEPARATE"  
(0040,a730) Content Sequence  
(ffff,e000) Item  
(0040,a010) Relationship Type "HAS OBS CONTEXT"  
(0040,a040) Value Type "PNAME"  
(0040,a043) Concept Name Code Sequence  
(ffff,e000) Item  
(0008,0100) Code Value "209069"  
(0008,0102) Coding Scheme Designator "99PMP"  
(0008,0104) Code Meaning "Observer"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0040,a123) Person Name "Clunie^David^A^Dr."  
(ffff,e00d) Item Delimitation Item  
... rest of content items go here...  
(ffff,e0dd) Sequence Delimitation Item
```



**FIGURE 35. Example of Inheritance of Observation Context**

---

## Inheritance of Observation Context

In the discussion of content and relationships in preceding sections, no mention has been made of any particular traversal order for the tree, nor of the concept of inheritance. For most content, no such notions are necessary to define the meaning, beyond the forward (top-down) definition of relationships like “contains,” “has properties” and “inferred from,” which clearly establish the source and target of the relationship. Even in the case of headings which are the concept names of container content items, the notion of containment is sufficient to establish a hierarchy of headings, contained sub-headings, and so on.

The one exception is observation context. It would be ridiculously repetitive to have to decorate every content item in the tree with its own complete observation context. It would probably be unsafe to leave propagation of observation context undefined or implicit, since different assumptions by different implementers could lead to ambiguity. Therefore rules for the explicit propagation of observation context are defined.

Specifically, observation context:

- ▶ is recursively inherited by a node from its immediate ancestor<sup>7</sup>
- ▶ is only propagated across by-value relationships, not by-reference relationships

This pattern of inheritance is illustrated in the example shown in Figure 35.

---

7. In the standard, a content item is said to “inherit the accumulated observation context of its parent.”

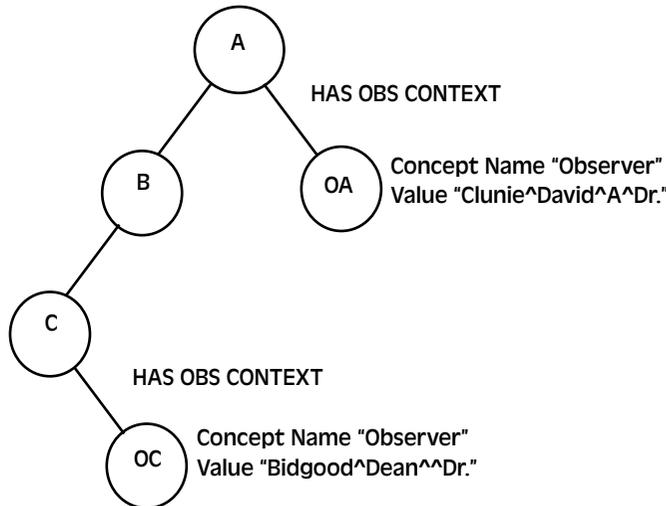


FIGURE 36. Replacement of Observation Context

**Not yet  
standardized.  
Final standard  
may differ  
substantially.**

Determination of the observation context of a particular node requires either a top-down traversal that accumulates the context, or a bottom up search to locate a parent with a particular attribute of context of interest.

Currently in the standard, it is stated that one can never replace an attribute of observation context lower down in the tree. For example, once the observer has been established,<sup>8</sup> none of that content item's descendants, no matter how distant in the tree, may ever redefine that attribute of observation context again. However, this rule has been found to cause difficulty, and no one can come up with a good reason for why it was introduced in the first place. Accordingly, it is very likely that this restriction will be removed, and replacement of attributes of context will be explicitly permitted.<sup>9</sup> In other words, the pattern illustrated in Figure 36 is currently explicitly forbidden but will very likely soon be permitted. Indeed it is expected to become a key feature of the observation context templates.<sup>10</sup>

In the majority of cases, most, if not all, observation context can be inserted high up in the tree as attributes of the root node (the document title). For example, in a simple chest X-ray report that does not quote content from prior reports (perhaps only referring to them), all the attributes necessary to describe the observer, the observation subject and the procedure can be attached to the root node. Indeed, almost all of

8. By decorating a content item with a "has observation context" relationship to a child with a concept name of "Observer."

9. One subtle concern is that observer, subject or procedure context might unintentionally only be "partially" overridden or replaced. It is proposed that this be expressly prohibited.

10. This is addressed in Supplement 53, the most recent draft of which at the time of writing was revision 21, dated October 3rd, 2000.

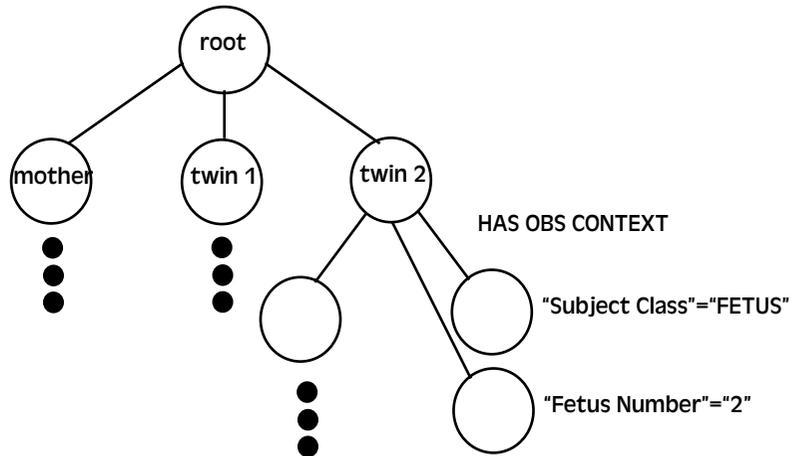


FIGURE 37. Observation Subject for Obstetric Ultrasound

these attributes can default to the values in the general modules of the composite IOD, without being explicitly redefined in the tree, as discussed in a later section.<sup>11</sup> Only the observer may need to be specified explicitly.

More complex cases arise when different parts of the tree require different values for attributes of observation context. Clearly, anything that is common can be factored out and inserted at a common ancestor. Anything that is different must be inserted on different branches of the tree. The obstetric ultrasound case of a multiple pregnancy mentioned earlier is an interesting example. Normally, one considers a procedure to have a single “observation subject.” In the case of a twin pregnancy, the “observation subject” may potentially be the mother, one twin or the other twin.

To define the mother, one might specify:

```

"Subject Class" = "Patient"
"Subject Name" = "Smith^Mary"

```

and to define the first twin, one might specify:

```

"Subject Class" = "Fetus"
"Fetus Number" = "1"

```

Using this approach, one clearly has to:

11. Notice that the standard explicitly specifies that observation context is first inherited from the general modules in the composite IOD, which are conceptually “above” the root node for the purpose of definition of inheritance. However, this initial (default) context may be overridden “if ambiguous.” What precisely “ambiguity” means is not further defined, but one must be prepared for any attribute of the initial context to be explicitly redefined, whether it be high up or towards the leaves of the tree.

- ▶ override the “default” observation subject inherited from the general modules of the composite IOD, which will almost certainly specify the mother as the patient
- ▶ be sure to construct completely separate branches of the tree for all the observations pertaining to the mother, the first fetus and the second fetus

Notice that, using strictly by value relationships, one really has to leave the observation subject unspecified for any attributes composed of information from multiple subjects. For example, one cannot specify the observation subject for a computed value of gestational age derived from attributes of both the first and the second twins’ measurements, or the mother’s last normal menstrual period.

There are two approaches to resolving this conflicting definition of context for such derived attributes:

- ▶ make use of by reference relationships
- ▶ “refine” rather than “redefine” attributes of context

An example of the first approach is to specify the derived gestational age as an attribute of the mother,<sup>12</sup> and to specify the fetal measurements from which it was derived using a by-reference “inferred from” relationship.<sup>13</sup>

An example of the alternative approach is to specify the observation subject at the top of the tree in very general terms, and to “refine” the specification lower in the tree. In the obstetric case, the entire tree might have an “observation subject” that is the mother:

```
“Subject Class” = “Patient”  
“Subject Name” = “Smith^Mary”
```

Further down the tree one might then refine the specification of observation subject to be one of the fetuses, for this purpose the fetus being considered a sub-part of the mother who remains the observation subject:

```
“Fetus Number” = “1”
```

Once one has refined the observation subject to specify the first twin, one can’t redefine the “fetus identifier” as the other twin down that branch. Notice also that one must not re-specify the “Subject Class” as “Fetus”, since that would redefine rather than refine the observation subject. Perhaps one could create an attribute like “Subject Sub-class” with a value of “Fetus”. This approach obviously doesn’t scale very well if successively more granular “refinement” of observation subject is required.

Hopefully, the current prohibition on redefining attributes of observation context will be lifted, and it will be possible to completely redefine the observation subject with

---

12. Which may be appropriate, since it is usually important in order to determine the mother’s estimated due date (EDD).

13. As it turns out, the “inferred from” relationship gets used by-reference a lot, but that does not mean that other types of relationship cannot be by-reference, nor that all “inferred from” relationships have to be by-reference.

the appropriate class and identification. If this proposed change in the standard is adopted, observations about the mother can have a context specified in the form:

```
"Subject Class" = "Patient"  
"Subject Name" = "Smith^Mary"
```

and observations about the fetus can have a context of:

```
"Subject Class" = "Fetus"  
"Fetus Number" = "1"  
"Mother of Fetus" = "Smith^Mary"
```

In the interim, the simplest approach is to have a single entity that is the observation subject, and not to "refine" it. For derived measurements from multiple subjects this requires the use of by-reference relationships, which in turn requires the use of the Comprehensive SR SOP class. It also creates a need for the creation of some kind of "composite" observation subject descriptions for derived observations on subjects that cannot neatly be categorized, such as "both twins" for example.<sup>14</sup>

## Initial Context

Since DICOM structured reports are always composite objects, there is already a considerable amount of observation context information present in the top level data set. In particular, the Patient, Specimen Identification, General Study, Patient Study and General Equipment modules already contain sufficient information to identify the subject and the procedure. In many cases, there is even information to identify the observer, particularly when the observer is a device and not a person.

This information from "outside" the content tree is referred to as "initial" context or "default" context. It need not be repeated in the tree. The standard states that the initial observation context from outside the tree can be explicitly replaced rather than inherited if it is ambiguous.

In the simple case of a human observer reporting on images of a patient that were acquired as part of the current study, all context can be inherited from the initial context, perhaps with the exception of the observer. The observer is a special case, since it is not obvious from which attribute outside the tree it could be inherited. A potential candidate is an optional attribute from the General Study Module, Name of Physician(s) Reading Study (0008,1060). Though this is not called out in the (informative) example of initial context in the standard, there is no reason why it cannot be used. Another potential candidate is the verifying observer specified in the SR Document General Module. However, it is probably not a good idea to elide explicit definition of the observer context on the assumption that the document will later be verified by the same person who authored report.

The precise mapping between attributes of initial context and corresponding attributes of explicit observation context will need to be defined in the templates for observation context.

---

14. Or what about "triplet 1 and 2" of three? This is obviously not as easy as it looks at first sight.

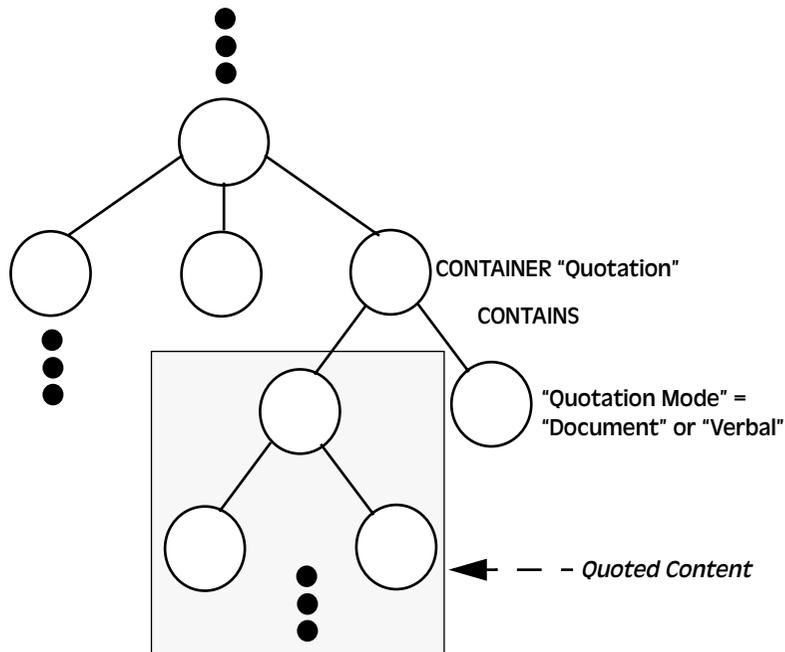


FIGURE 38. Quoted Content and Quotation Mode

**Not yet standardized. Final standard may differ substantially.**

## Quoted Content

Up to this point, the assumption has been that the context of the observations is “direct.” That is, the creator of the SR document (the observer) has been creating content items de novo that directly describe the subject. This is not always the case. Sometimes the creator of the document needs to quote either another document (such as a prior report), or something that was told to them (such as the history as spoken by a patient). Furthermore, quoted content may itself quote other material, and so on, recursively.

There are no “hard-wired” quotation capabilities in the standard. Rather this is left up to templates and observation context. What follows are a number of possibilities for quoting content, including a description of the most recent proposal,<sup>15</sup> but remember that this is an area that is in a state of flux.

The following requirements are common to quoted documents and speech:

- ▶ the point in the tree at which quoting begins must be distinguished
- ▶ the person or device doing the quoting must be identified

<sup>15</sup> From Supplement 53, the most recent draft of which at the time of writing was revision 21, dated October 3rd, 2000.

- ▶ the person or device being quoted must be identified

In the case of a quoted document it is also necessary to be able to identify the document itself.

Earlier proposals for implementing templates for quotation suggested explicitly named concepts for these requirements, such as “quoted author,” “quoted person” and so on. However, quoted material is not significantly different from other content, once it has been established that quoting has begun. Just like any other content, quoted content requires observer, subject and procedure context. For example, the author of material that is being quoted, whether it be a document or spoken, is in essence an “observer” like any other. Furthermore, quoted content is always effectively a “leaf” of the current document. That is, once one starts quoting, any further descendants will be from the quoted document (or one that it in turn quotes).

Accordingly, one can meet the requirements for quoted content by:

- ▶ specifying that the context “above” the quote describes who is making the quotation and in what context
- ▶ explicitly specifying the point that the quote begins, with a container with a concept name of “Quotation”
- ▶ including within that container the quoted content, together with its own observation context that identifies who or what is being quoted and in what context the quoted content was created

The “Quotation” container can contain additional attributes to identify the document being quoted, and the mode of quotation (whether it is a document or verbal), as illustrated in Figure 38.

This approach satisfies the requirement to support recursively quoted content and avoids a proliferation of specific attributes to described quoted content. It does depend on the ability to replace attributes of observation context, however.<sup>16</sup>

As an example of quoting a document, consider describing in a report the progress of a mass in the brain as seen on MRI. A prior report could simply be referenced by using the COMPOSITE Value Type, and explaining or paraphrasing the reference:

```
<TEXT:(,,“Finding”)=“The 5cm mass described in”>
<COMPOSITE:(,,“Prior report”)=(“SR Text”,“1.2.3.4...”)>
<TEXT:(,,“Finding”)=“has enlarged and now measures 7cm”>
```

By using a quoted document, some of the findings from the prior report can be directly included in the current report:

```
<TEXT:(,,“Finding”)=“The mass previously described as”>
<CONTAINER:(,,“Quotation”)>
  <contains CODE:(,,“Quotation Mode”)=(,,“Document”)>
```

---

16. It turns out that this is also true of earlier proposals, and the current prohibition of replacement of attributes of observation context was apparently not intended to apply if the “mode” of observation were changed from direct to quoted.

```
<contains CONTAINER:(,,,"Quoted Content")>
  <has obs context COMPOSITE:(,,,"Quoted Source")=...>
  <has obs context PNAME:(,,,"Observer")="Smith^Z^^Dr">
  <has obs context UIDREF:
    (,,,"Procedure Study Instance UID")="5.6.7.8">
  <contains TEXT:(,,,"Finding")="5cm diameter enhancing
    mass in the right frontal region">
<TEXT:(,,,"Finding")="has enlarged and now measures 7cm">
```

Notice in this example that:

- ▶ the observer in the tree above the quotation is the person doing the quoting
- ▶ the observer context of the quoted content is explicitly redefined, such that the author of the previous report is identified using the observer concept
- ▶ the procedure context of the quoted content is explicitly redefined, such that the procedure that was the subject of the previous report is identified by its Study Instance UID
- ▶ the subject context is not redefined, and hence is inherited; i.e. the prior report being quoted and the current report are both about the same patient
- ▶ the quoted content is established to be from a document rather than a verbal statement by the value of "quotation mode"
- ▶ the actual document that is being quoted (the prior report) is explicitly identified by the "quoted source" attribute

Care needs to be taken when quoting a fragment of another document to capture the accumulated context at the root of the fragment. When a fragment is copied from its original tree, much if not all of its observation context may be lost unless the accumulated inherited context is also copied and attached to the root of the quoted fragment. In the example, the observation context would very likely have been attached to the root node of the entire document, and yet only one text content item is being quoted.

As was mentioned, this approach depends on observation context being overridden in quotations. It is important to ensure that inappropriate context of the current document is not inherited by the quoted content as a consequence of the nesting.<sup>17</sup> For this reason it is expected that templates for observation context will specify that whenever any attribute of a category of observation context (observer, subject or procedure) is overridden or replaced, all other attributes of that category (only) are removed rather than inherited.<sup>18</sup>

---

17. This could occur if some optional attribute of context was specified in the current document but was left unspecified in the quoted content. For example, a surgical pathology report might record a "Specimen Identifier" in the current report, and then quote a report on a previous specimen that did not mention such an identifier, only an accession number. Obviously the current identifier should not be inherited by the quoted content.

18. Indeed, while considering this issue it became apparent that there is no way to specify a null value for an attribute during replacement or overriding, since most content items are not allowed to have zero length values.

## Not Observation Context

The standard explicitly mentions the attribute Observation DateTime (0040,A032) as being conditionally present in every content item, if different from its parents or the document as a whole.<sup>19</sup> This is a form of “inheritance” defined explicitly for this attribute, outside the scope of observation context per se. Indeed, the sub-title of this chapter is something of a misnomer, since “when” an observation was made is not really part of “observation context” at all.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ observation context has three categories: observer, subject, and procedure
- ▶ observation context is encoded in the same way as any other content, but uses a HAS OBS CONTEXT relationship to attribute it to a parent
- ▶ observation context is propagated top-down, that is the accumulated context of the parent is inherited by all its children
- ▶ whether attributes of observation context may be overridden or not is an open issue; in the present standard it is forbidden, but this is expected to change
- ▶ initial (default) context is inherited from the general modules of the composite object
- ▶ initial context may be redefined (overridden) in the tree if “ambiguous”
- ▶ documents and speech may be included as quoted content, potentially recursively
- ▶ date and time of an observation are “inherited” but are not part of observation context

---

<sup>19</sup>. Specifically, it needs to be present if different from Content Date (0008,0023) and Content Time (0008,0033), formerly known as Image Date and Image Time, respectively.



# ***Document Management: SR in the Real World***

---

There is more to structured reporting than just creating content and encoding it. Once created, documents must be managed such that they are stored and available where needed. Specifically one needs to understand such things as:

- ▶ from whence documents come
- ▶ to where documents go
- ▶ what version of a document is in use
- ▶ how to find the relevant document

## **Instances**

What is this thing called an “instance?” In object-oriented terminology, one distinguishes the notions of a description of an object, its “class,” and an actual occurrence or “instance” of an object. As is described in the chapter on “SOP Classes and Conformance,” DICOM uses similar terminology. A SOP<sup>1</sup> Class pairs the description of the data (an IOD<sup>2</sup>) with a service such as network or media storage, and a SOP Instance is an actual occurrence of a data set. This chapter is all about instances; how instances come into existence, where they are consumed, where they are stored, and how they are found.

## **Unique Identifiers**

Every instance of a document is assigned a “unique identifier” (UID) which is different from the UID of any other document. This is achieved by a hierarchical naming

---

1. SOP: Service-Object Pair  
2. IOD: Information Object Definition

scheme that is defined by a central authority.<sup>3</sup> The primary authority assigns a “root” to a secondary authority and delegates responsibility for managing all UIDs starting with that root. The secondary authority in turn delegates responsibility to tertiary authorities and so on recursively. The result is a string composed of numeric components separated by periods. For example:

- ▶ ISO is assigned the root “1”
- ▶ ISO delegates responsibility to its internal member body branch, which is assigned (by ISO) the root “1.2”
- ▶ ISO’s member body branch delegates responsibility to the US member body, ANSI, which is assigned (by the ISO member body branch) the root “1.2.840”
- ▶ ANSI delegates responsibility to organizations that register with it, such as NEMA, which is assigned (by ANSI) the root “1.2.840.10008”<sup>4</sup>

Another example is one of the author’s own roots:

- ▶ ISO is assigned the root “1”
- ▶ ISO delegates responsibility to its internal member body branch, which is assigned (by ISO) the root “1.2”
- ▶ ISO’s member body branch delegates responsibility to the US member body, ANSI, which is assigned (by the ISO member body branch) the root “1.2.840”
- ▶ ANSI delegates responsibility to organizations that register with it, such as General Electric, which is assigned (by ANSI) the root “1.2.840.113619”
- ▶ GE delegates responsibility to an internal group that manages UID roots for DICOM products, which is assigned (by GE), the root “1.2.840.113619.6”
- ▶ the internal GE group delegates responsibility to the author for the root “1.2.840.113619.6.48”

In this last example, this means that the author can create any UID that begins with “1.2.840.113619.6.48”, and be assured that no-one else on the planet (barring an error or deliberate abuse) will create UIDs that begin with the same string. It also means that the owner of the root is responsible for managing this assigned “space,” and making sure that no two devices using this space create duplicate UIDs. There are many ways to achieve this, by such techniques as:

- ▶ sequentially assigning portions of the space to different product families
- ▶ further sub-dividing the space by purpose, such as for SOP Instance UIDs, private SOP Class UIDs, Implementation Class UIDs and so on
- ▶ yet further sub-dividing by device serial number

---

3. The rules are specified in an ISO standard, ISO 9834-3.

4. UIDs are used in DICOM for many purposes, not only to identify SOP Instances that are defined in the standard itself (“well-known SOP Instances”) but also to identify SOP classes and transfer syntaxes. Obviously, an instance of a DICOM object created by a device will never start with the DICOM assigned root “1.2.840.10008”, but rather with the device’s own assigned root.

- ▶ for each created instance on a particular device, sub-dividing by date, time, process number or some hashed value<sup>5</sup> that is a derivative thereof

and so on. Implementers should take care to do this well, since managing instances of objects depends on the uniqueness of UIDs.<sup>6</sup>

A few more points about UIDs:

- ▶ even if one knows how a device “constructs” UIDs, they should never be parsed to recover embedded information; UIDs are unique, that is all one can depend on; one must not try to find “hidden” study IDs, series IDs and so on buried inside UIDs
- ▶ since UIDs are globally unique, instances can be shared beyond the confines of a single organization and never conflict with another object; one must never use the same root in two organizations in the hope that they will never be connected

This discussion of what is a fairly low level technical detail may seem premature or tedious, but is placed here to emphasize that Structured Reports, like all other objects (instances) in DICOM are uniquely identifiable, wherever they may end up “in the system.” As such, SOP Instance UIDs are the key to managing, finding, matching and especially cross-referencing content in other reports and objects. In earlier chapters, references to images, waveforms and other composite objects in IMAGE, WAVEFORM and COMPOSITE value types have been described, each of which uses SOP Instance UIDs to specify the target of the reference.

### “Document” Paradigm: Persistence vs. Transience

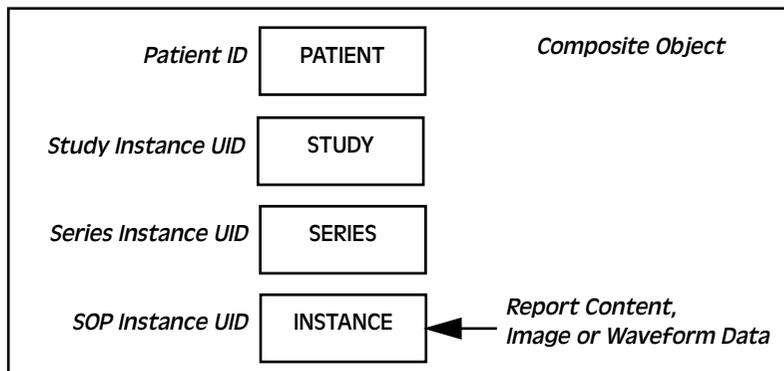
DICOM objects of all kinds have a “persistence” beyond a single transaction. This is referred to as a “document” paradigm, as compared to a “message paradigm.” By contrast, for example, HL7 uses a “message paradigm:” HL7 messages are constructed, sent, consumed, and then cease to exist; they are transient. While the receiver of a message may extract information and do something with it to cause it to remain available, such as insert it in a database, the message itself does not persist (except perhaps as an entry in a log, journal or audit trail).

Composite DICOM objects can be used in the same way if so desired. There is no reason why a device that is sending an image or report cannot construct a DICOM object from an internal store, assign it a brand new UID, and store it across the network to a device that deconstructs the object, stores bits and pieces of it in a database, and discards all the UIDs. Indeed, that is a fairly common and perfectly valid pattern of use, especially when the receiving device does no more than render the object for display, without keeping any persistent state at the receiving end once the user is done.

---

5. UIDs have a 64 character limit so care needs to be taken if one gets too “creative” that this limit is not exceeded.

6. Implementers can obtain a suitable “root” for generating UIDs from a variety of sources. See “<http://idt.net/~dclunie/medical-image-faq/html/part8.html#UIDRegistration>” for a list. Note that one is not required to use a root from the ISO national member body of one’s own country.



**FIGURE 39. Composite Objects Represent Multiple Entities**

More commonly, however, DICOM composite objects are treated as if they were “documents,” just like paper documents. They are brought into existence, they persist, they are archived for subsequent retrieval, they may cross-reference each other, copies and new versions of them may be made, and so on. DICOM provides services not only for storing persistent objects across the network and on media, but also services to search (query) for them, and retrieve them, as will be discussed later in this chapter. The scenario most commonly envisaged is an archive that acts as central repository of objects, is a sink for creators of objects, and serves a source that may be queried and from which selected objects may be retrieved.

Every structured report is a composite object. Like all other composite objects it contains information that identifies and describes:

- ▶ the patient (or specimen)<sup>7</sup>
- ▶ the study
- ▶ the series<sup>8</sup>
- ▶ the instance itself, including the document content

See Figure 39. A composite object document is largely self-contained and self-describing. It can be meaningfully used and displayed without consulting any other source of information about, say, the patient. The same applies to composite objects that are not reports, such as images or waveforms, each of which is also usable in its own right.

For the document paradigm to be successful, particularly in a heterogeneous multi-vendor environment, all the devices involved have to share certain assumptions and conform to some ground-rules. Though DICOM stops short of defining an entire

7. Or in a more general sense, the “subject.”

8. The series level has little meaning in structured reporting, specifying little more than the modality (“SR”) and the identification and description of the equipment.

architecture for managing reports, images, waveforms and other composite object instances, it does specify that:

- ▶ all instances are uniquely identified, and that unique identifiers may not be reused
- ▶ any changes that are made to the contents of an instance that are substantive must result in a new instance

The latter point is so important that it deserves further elaboration.

## Copies and Versions

DICOM image objects are defined such that if the pixel data of a derived image is different from the pixel data of a source image, and this difference is expected to affect professional interpretation of the image, the derived image shall have a different UID. This is a pretty vague statement, and has led to a lot of problems managing images. Some vendors interpret this statement as giving them the freedom to change “minor” details stored in a composite image, such as the window center and width information, without changing the SOP Instance UID, and to “re-send” the image, for example, from the acquisition modality. This practice creates the risk that if the image has been sent to a number of different places (such as a central repository, a long term archive, and/or multiple review workstations) then inconsistencies will arise amongst objects that ostensibly have the same SOP Instance UID. Even worse, someone might “correct the patient’s name,” for example in the event of a misspelling or a trauma case, at a peripheral station, and resend the image with the same SOP Instance UID, hoping that the change will propagate.

The behavior of a receiving device, on receiving the same SOP Instance UID twice is undefined. It may discard the “duplicate,” it may replace its existing “version” with the “duplicate,” or it may be very smart, not trust anybody, and comparing the “new version” with the “old version,” detect a difference, and perform some appropriate action. In an ideal world, the appropriate action might be to update the database of identifying information, propagate the changes to all peripheral devices that have “stale” versions, as well as to flag the old version on the long term archive as invalid and queue the replacement version for archiving again to long term storage. This behavior implies a lot of centralized intelligence and control over management of objects, which doesn’t exist in all systems. The problem is even worse for images stored on interchange media, since such media are often used in a “write once” manner, not allowing for any change short of replacement of the piece of media entirely.

At the present time, since the behavior of a device in this respect is unpredictable, one should avoid at all costs making changes to image objects with the same UID, and depending upon the consequences.

For reports, one must be able to do better. Reports are much more likely to be “living documents” in the sense that their content will evolve and be maintained. For example, a report might be created as an initial draft, reviewed, corrected and amended, after which it will be verified as being ready for distribution. One can argue that the report shouldn’t be “in the system” until after it is verified, but in reality it is common practice to make “interim reports” available to users, flagged as preliminary,

in order to manage the patient's problems in a timely manner. In some environments it takes considerable time (often up to 24 hours) to complete and verify a preliminary report, by which time it may be too late to affect a patient's care.

Furthermore, even completed and verified reports may evolve and be amended, as more information comes to light (such as old exams becoming available, or more clinical information being presented). It is not sufficient to simply change the contents of a report, or create a new one. It is vital to be able to record the fact that amendments have been made, and preferable to be able to reference the previous versions.

All these requirements may seem reasonable, if not obvious, yet it is difficult to write hard and fast rules that are appropriate in every situation. Ideally one could design an entire system or architecture, including various states that a report could be in, together with rules for transitions between states. However, reaching consensus on such matters between multiple vendors with competing ideas and concerns is not realistic<sup>9</sup>. Also, experience has shown that inside a composite object is not necessarily the best place to embed state information, which is probably better stored "out of band" inside the management system.

Yet one can't just forget all about this, and leave it to the vendor's discretion, and hope to achieve interoperability. Accordingly, the composite SR document contains the following "management" information:

- ▶ a SOP Instance UID, which (as for all composite objects) identifies the instance
- ▶ strict rules for when attributes of the object may be changed without changing the SOP Instance UID
- ▶ indication of whether the document is "partial" or "complete"
- ▶ indication of whether the document is "verified" or "unverified"
- ▶ if verified, who verified the document and when
- ▶ when the content of the document was created
- ▶ lists of predecessor and identical documents
- ▶ lists of current and prior evidence (e.g. images used to prepare the report)
- ▶ information about the request and the procedure performed

Most of these will be discussed in later sections of this chapter. For now, how this information is relevant to various different "versions" of a structured report instance will be discussed.

The rules for when to change the SOP Instance UID are:

- ▶ addition, removal or update of any attribute within the "management information" of the document (specifically, the SR Document General Module)

---

9. Some of the "more interesting" discussions at WG 6 with respect to structured reporting involved the matter of encoding of states and management information. "More interesting" in this context is used to mean much wailing and gnashing of teeth, not to mention shouting and banging of fists on the conference table.

- ▶ addition, removal or update of any attribute within the “content tree” (specifically, the SR Document Content Module)
- ▶ modification of the Series Instance UID (0020,000E)
- ▶ modification of the Study Instance UID (0020,000D)

In particular, one must create a new instance if:

- ▶ the state of the document is changed (e.g. an unverified report is verified)
- ▶ the content of the document is changed (e.g. a finding or recommendation within the content tree is amended)
- ▶ the document is moved or copied into a new study or series

Notice however that there are some situations in which the DICOM SR object may be altered without changing the SOP Instance UID:

- ▶ the patient’s name and identifying information could be changed, since that is stored in the General Patient Module and not the SR Document General Module
- ▶ information describing the study that the report is part of may be updated (with the one exception of the Study Instance UID), since that is stored in the General Study Module and not the SR Document General Module

Since the information about whether a document is partial or complete, and verified or unverified, is stored in the SR Document General Module, then any change in “state” implied by those attributes *must result in a new SOP Instance UID*. This is a crucial point, since it means that finally a line has been drawn that says more or less that “new versions of an SR document must have a new SOP Instance UID”. Given this rule it now becomes possible to:

- ▶ reference a particular version of a document by its SOP Instance UID without risk that it will point to the “wrong” version
- ▶ reference prior versions of a document from within a later version of a document
- ▶ quote the content of a particular version of a document identified by its SOP Instance UID, without fear that the version of the document quoted will “evolve” after it has been quoted

Indeed, not only is it possible to identify a quoted or predecessor document, but it is actually required that the Predecessor Documents Sequence (0040,A360) be present and contain such references. This is illustrated in the Figure 40.

So far, different “versions” of a document that might come into existence as a result of verification, changes and amendments have been discussed. Not yet discussed is the notion of “copies.” How could a “copy” possibly arise, one might ask, given that a single DICOM composite instance can be distributed anywhere and have many identical copies in multiple locations?

The issue arises in a different context, that is, when the same document needs to be part of two different studies, and consequently requires different identifying information. If, for example, a single report is generated in response to two different requested procedures, then that report needs to become attached to both those requests, and the products (such as images) of both those procedures. Cross-refer-

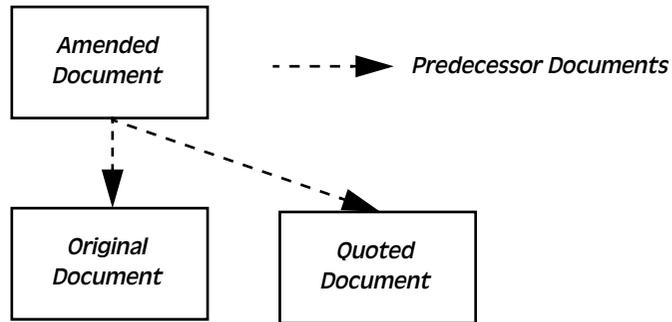


FIGURE 40. Predecessor Documents Sequence

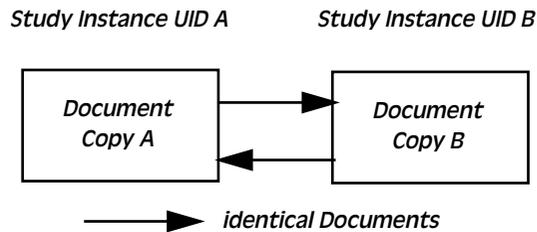


FIGURE 41. Identical Documents Sequence

encing everything would be one potential solution, were it not for the fact that the DICOM composite information model is strictly hierarchical. Specifically, the same study entity can only be contained within a single patient, the same series entity can only be contained within a single study, and the same instance can only be contained within a single series. In other words, an object with the same SOP Instance UID cannot be contained in two different studies.<sup>10</sup>

One possible solution to this problem is to duplicate the entire document, except for the study identifying information, and assign new series and instance UIDs. If this approach is taken, then the standard requires that the two copies cross-reference each other. Specifically, it requires that in both documents, the Identical Documents Sequence (0040,A525) contain the SOP Instance UID of the other copy (or copies), as illustrated in Figure 41.

One of the reasons for this requirement is so that if a document is later amended, then the presence of an Identical Documents Sequence will alert the amending system that it has to seek out and amend the other copies as well. In that situation, new

10. It might appear as though the Referenced Study Sequence attribute in composite objects allows this, but actually all that sequence contains is potential aliases for the same study, which arise as a consequence of objects being created when not connected to a management system (such as via modality worklist), and subsequently being assigned a new Study Instance UID.

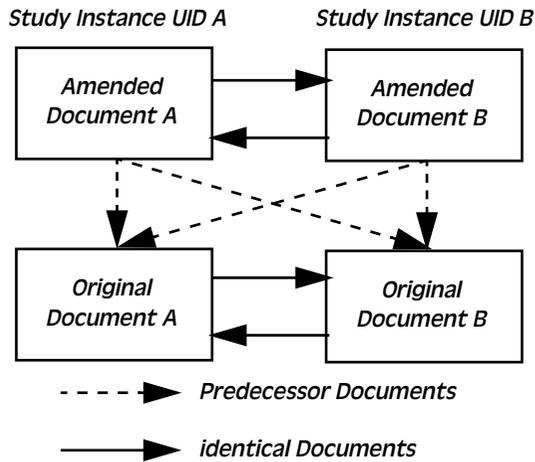


FIGURE 42. Identical Documents with Predecessors

amended copies will be made as well, each cross-referencing the other, as well as each pointing to all the predecessor documents, including all the previous copies.<sup>11</sup> This rather complex web of cross-references is illustrated in the Figure 42.<sup>12</sup>

### States: Completion and Verification

Up to this point, terms like “completed,” “verified” and “amended” have been used without precise definition. Different versions of documents may be linked together by UID references, but it remains to be discussed how and under what circumstances different versions come into existence. As has been mentioned, the definition of “states” for documents, and in particular reports, causes controversy, straying as it does into the territory of system design and architecture, territory that is taboo for DICOM.

The minimal level of state information permitted inside a structured report consists of:

- ▶ indication of whether the document is “verified” or “unverified”
- ▶ indication of whether the document is “partial” or “complete”

In earlier drafts there was also an “amended” state that was subsequently dropped.

The verified state is the easiest to define, since everyone seems to share a pretty clear understanding that at some point in the history of a document, someone will stand up and take responsibility and say it is ready for popular consumption. In the standard,

11. This requirement is expressly stated in the standard. It is not allowed to create successor duplicates that each only point to their “own” predecessor.

12. The idea for this illustration comes from the IHE Year 2 Technical Framework, which contains an even more complex example illustrating the case of three copies.

the expression that is used is “accountable for its content.” Just as a paper report will ultimately receive the author’s hand-written signature, or the signature of someone else with the authority to verify their work (such as a superior, peer or replacement), so does an electronic report need a similar attestation.<sup>13</sup> A hand-written signature serves both to indicate that a report is verified as well as to uniquely and in a non-repudiable<sup>14</sup> manner identify the verifier. In DICOM SR, the statement that the document has been verified, and by whom, is contained within the document. However, the equivalent of a unique and non-repudiable signature probably requires an additional cryptographic digital signature. Digital signatures are discussed in the chapter on “Security: Privacy and Authenticity.”

Whether or not a document is verified is specified in the mandatory attribute Verification Flag (0040,A493), which may contain either “VERIFIED” or “UNVERIFIED”. A sequence of items in Verifying Observer Sequence (0040,A073) contains information about:

- ▶ when the document was verified, in Verification DateTime (0040,A030)
- ▶ who verified the document, in Verifying Observer Name (0040,A075)
- ▶ to what organization they belong, in Verifying Organization (0040,A027)

If the verifying observer<sup>15</sup> also has a coded identifier, such as a national provider identifier,<sup>16</sup> that may also be recorded, in Verifying Observer Identification Code Sequence (0040,A088).

There may be more than one verifying observer,<sup>17</sup> and each may perform the verification at a different date and time. This is controversial. Some people believe that one and only one individual should take responsibility for the content of a document. In reality, of course, people like to spread responsibility about, especially when at risk for litigation. There are two common scenarios in which multiple verifying observers may be appropriate:

- ▶ when both a trainee and their supervisor (e.g. resident, registrar or fellow and staff physician or consultant) both verify a report, perhaps sequentially or perhaps simultaneously

---

13. “Attestation” is one of those horrible legal words that has crept into medical jargon. It is used to mean “confirmation” or “verification” by someone who will take responsibility for the content.

14. “Non-repudiable” in this context means one can’t deny later that one did sign (and additionally in some contexts, the date and time the signature was placed).

15. Notice the subtle switch to referring to the person verifying the document as the “verifying observer.” The word “observer” has crept into the standard with Supplement 23. There are various types of observer, including the “verifying observer” described here, as well as other types of “observers,” whose roles are defined in the chapter on “Context: Who did what to whom, when...”. For normal people, an “observer” is pretty much the same as an “author.”

16. Unlike normal people who live in civilized countries and have rights to privacy and anonymity, physicians and other health-care providers aren’t granted the same privilege, on the grounds of public policy. Hence they are assigned identifiers by a central authority. They are not actually required to be tattooed with bar-codes or have chips implanted, at least not yet.

17. Which is why a sequence of items is used.

- ▶ when two responsible parties contribute equally, for example when “double-reading” a screening mammogram

The bottom line here is that it is not DICOM’s responsibility to adjudicate on who should or should not be ultimately responsible for a document. That is for the law, the courts and standards of professional practice to determine. Regardless, DICOM SR provides a place to encode one or more verifying observers to whatever extent site policy and local law require. Implementers, especially those designing receivers of documents, should take care to allow for the possibility of more than one verifying observer in their databases and user interfaces.

The completed state is not so easy to define. In a general sense, most people can agree that a document is “complete” when it is finished with respect to whatever its purpose is. This of course begs the question as to what is that purpose.

If a document is a traditional report, then it is complete when no further amendments are anticipated and it is ready to be released for public consumption, without being flagged in some way as “interim.”

On the other hand, if a draft of a report is being prepared and work is temporarily suspend, then later resumed, then ultimately that draft may be “completed.” The draft may subsequently need to be revised and amended before being verified and released, but from the perspective of it being the “entire” report as opposed to just a piece of it, it is still “complete” for this purpose.

Another example is an intermediate product, for example measurements made on an acquisition device. From one perspective, the measurements encoded in a structured report object are complete when the exam is done, the patient has gone home, and the operator has signed the case out. From another perspective, that of the radiologist’s report on the procedure, the measurement SR document is a partial report that contains results that are destined to be incorporated in their more complete report.

In the last case, should the acquisition device flag the report as partial or complete? How is the acquisition device to magically sense the subsequent intent of the consumers of the document? Also, the document will very likely also be verified, by the operator.

Regardless of how meaningless the concept of “completeness” might be without knowledge of the purpose for which the document is used, it is sufficiently useful for many common applications that a compromise had to be reached. The following rules apply:

- ▶ the mandatory Completion Flag (0040,A491) is defined to mean “the estimated degree of completeness of this SR Document with respect to externally defined criteria in a manner specified in the Conformance Statement,” and has values of “PARTIAL” or “COMPLETE”
- ▶ an explanation of the value may optionally be sent in Completion Flag Description (0040,A492)
- ▶ the date and time that the document creation started is required to be sent in Content Date (0008,0023) and Content Time (0008,0033)<sup>18</sup>

- ▶ the date and time of document verification are specified in Verification DateTime (0040,A030), as described earlier
- ▶ the “prevailing final version” of an SR Document is defined to be the version having the most recent Verification DateTime (0040,A030),<sup>19</sup> Completion Flag (0040,A491) of COMPLETE and Verification Flag (0040,A493) of VERIFIED

There are a number of implications of these definitions. For a start, it leaves the discretion for use and definition of the completion flag up to the implementer. In the real-world this translates into “do not depend on this flag” if there is to be any hope of interoperability. In particular, one must be wary of keying off pseudo-defined terms that may be passed in Completion Flag Description.<sup>20</sup>

The bottom line here is that:

- ▶ to convey the notion that a document is somehow “finished,” it needs to be both complete and verified
- ▶ if such a finished document is subsequently amended, then the final form of the amendment also needs to be both complete and verified, and it will be distinguishable from its predecessor by a more recent verification date-time
- ▶ drafts of amendments in progress may be “complete” but should not be “verified”
- ▶ a “complete” but unverified document may be an “interim,” “preliminary” or “draft” of some kind, distinguishable perhaps from a “partial” report which is just a piece of a document, not an entire document

Notice especially that the presence of the verified flag does not necessarily imply that the document is a “final” document. For example, a report may be verified, then later amended, in which case it will likely be verified again, possibly by someone else. Indeed, intermediate products destined for inclusion in a higher level report may well be “verified” by the operator responsible for creating them, without there being any intent that the document stand alone as a “complete” document.

Finally, the use of the word “state” is probably inappropriate in this context, though it has been used here for clarity. The verification and completion flags are really only descriptors. In particular, there is no express or implied “state machine” in the standard for these attributes, and no transitions between states defined.

## **Current and Other Evidence: References Revisited**

The SR Document General Module also contains information about what “evidence” was used in the creation of a document, as well as “other” evidence that may be of rel-

---

18. These are actually the same attributes formerly referred to as Image Date and Image Time, in order to take advantage of existing support for these attributes during queries. Otherwise a new attribute with a DT VR would have been created.

19. Presumably the most recent of several Verification DateTime values if there is more than one item in the Verifying Observer Sequence within which this attribute is encoded.

20. For DICOM print aficionados, just think of this attribute as being as dangerous as Configuration Information. “Dangerous” in this context is used to mean “a threat to interoperability.”

evance. The word “evidence” is used as a generic term to refer to things like images, waveforms, measurements and reports used while making the “observations” that are encoded in the document. The evidence in question is stored in DICOM composite objects, which are referred to in the usual manner by their SOP Instance UIDs.

There are two mandatory sequences encoded in the SR Document General Module:

- ▶ Current Requested Procedure Evidence Sequence (0040,A375), in which is listed all the evidence that forms the “current” (interpretation) procedure
- ▶ Pertinent Other Evidence Sequence (0040,A385), in which is listed other evidence, such as prior images and reports

These are mandatory in the sense that they must be present if such evidence was used and it exists in the DICOM domain. Obviously if the document is a result of evidence that doesn’t exist as DICOM composite objects, then that evidence cannot be listed.

The concept of a “current requested procedure” needs to be further described. The paradigm here is that nothing is done in the real world except in response to a request.<sup>21</sup> This is because in many jurisdictions that would be illegal, and more importantly, one wouldn’t get paid for doing it. In this paradigm, a procedure is therefore always requested, then performed and then reported upon. Even if an SR document encodes an intermediate set of results rather than a diagnostic report, it is very likely still generated in response to a request of some sort.

From the perspective of encoding a DICOM Composite object to hold the structured report, this means that all the study identifying attributes, particularly the Study Instance UID, will be the same for both the object containing the report, *and the objects containing the images and other evidence. on which it is based.*

All of this evidence needs to be listed in the Current Requested Procedure Evidence Sequence, which serves a number of purposes:

- ▶ it provides resources to locate (and possibly pre-load or pre-fetch) instances of evidence that might be referred to within the body, for example in an IMAGE, WAVEFORM or COMPOSITE value type
- ▶ it provides a complete list of all the evidence that was available to the creator of the report (whether they actually looked at it or not)
- ▶ by its absence from the list, information subsequently added to the study after the report was created (such as further images acquired or reconstructed) can be determined not to have been available to the creator of the report

These last two purposes are a source of both liability and protection for the report creator, who cannot subsequently deny what evidence was available to them, but can subsequently verify what evidence was not.

---

21. *Except in the “trauma” or “emergency” scenario, in which case a post hoc request is usually artificially generated. This reduces the awkwardness and inconvenience that is a consequence of actually having to treat the patient, rather than the information system.*

The more pragmatic consequence of listing all the evidence is that references to composite objects can be fully elucidated in one place without ridiculously bloating the encoded document. Traditionally in DICOM, composite objects have been referred to by a SOP Class and SOP Instance UID pair. Unfortunately this is of very little help in actually tracking down and retrieving the instance, since most query-retrieve servers only support the hierarchical rather than relational models, and one has to know the Study and Series Instance UIDs before one can even perform a query on the SOP Instance UID.<sup>22</sup> To help users of SR documents, this problem has been resolved by:

- ▶ encoding the complete hierarchy of Study and Series UIDs as well as the SOP Instance UIDs in the evidence sequences of the SR Document General Module
- ▶ referring only to the SOP Class and Instance UIDs from within the SR tree<sup>23</sup>

In other words, the SR Document General Module contains a “cache” or “library” or “repository” that can be used to make the task of following actual composite object references in the tree easier. By encoding the evidence sequences as a nested hierarchy of study, series and instance, the representation is more compact than a list of every instance, each annotated with its study and series identifying information.

It is crucial for implementers to remember however, that the evidence sequences contain not only material that is referenced in the tree, but all evidence that was available to the document creator. *Failure to comply with this requirement may have product liability implications.*

The actual encoding of both evidence sequences, the Current Requested Procedure Evidence Sequence and the Pertinent Other Evidence Sequence, makes use of yet another macro, the SOP Instance Reference Macro, which includes the following study level information:

- ▶ the Study Instance UID (0020,000D)
- ▶ a Referenced Series Sequence (0008,1115)

in which is nested the following series level information:

- ▶ the Series Instance UID (0020,000E)
- ▶ optionally, the Retrieve AE Title (0008,0054) and Storage Media File-Set ID (0088,0130) and UID (0088,0140)
- ▶ a Referenced SOP Sequence (0008,1199)<sup>24</sup>

in which is nested the following instance level information:

- ▶ the Referenced SOP Class UID (0008,1150)
- ▶ the Referenced SOP Instance UID (0008,1155)

---

22. This will be explained in a later section in this chapter on the DICOM Query-Retrieve service.

23. As it happens, the SOP Class UID is specified both within the reference in the tree and in the SR Document General Module. There is no particularly good reason for this, other than consistency with all the other object references in DICOM.

24. Notice that this usage differs from the (unfortunate) historical convention of having different reference attributes for images as opposed to other objects.

Notice that only the UIDs are specified, there is no other more generic, human readable identifying information.

Be extremely wary of depending on the Retrieve AE Title. It is extremely bad design to assume that an object will stay put wherever it happens to be, or that there are not other more appropriate places where it might be found. For example, a creator of a document might place its own location on the network in Retrieve AE Title, only to delete the referenced objects, once they have been safely sent to an archive device. At this point, the referenced UIDs within the document remain valid, but the Retrieve AE Title is stale and useless. Furthermore, even if the document creator does still retain the referenced objects, it is probably inappropriate use of resources and network topology for a reader to retrieve them from the creator rather than an archive.

The moral of the story is that report creators should avoid putting their own AE Title in Retrieve AE Title, and indeed only fill this attribute if they can identify a sensible repository that is likely to be capable of serving up the referenced objects over the long term. When in doubt leave this attribute empty.

What is encoded in the Current Requested Procedure Evidence Sequence has been discussed at length; what about the Pertinent Other Evidence Sequence?

The primary reason for the pertinent other evidence list is to provide a repository to consult when referencing prior reports, or objects generated by prior procedures. It is very common practice to reference prior work from the body of a report of any kind. A typical statement is “compared to the previous exam ...” or “it was stated in the prior report that ...” and so on. One of the features of structured reporting is the ability to actually embed references to priors that can be followed, rather than just describing them as is done in a traditional printed report.

The objects listed in the pertinent other evidence sequence must be from a different procedure (i.e. have a different Study Instance UID than that of the document and the current evidence). Obviously the same objects may not be listed in both the current and pertinent other evidence sequences.

How much should be listed in the pertinent other evidence sequence? The rule that the current evidence sequence should contain all evidence examined (not just that referenced from tree) does not apply to pertinent other evidence. The definition of “pertinent” is somewhat wishy-washy, so it doesn’t make a lot of sense to mandate that it be fully elucidated. Actually, the standard doesn’t even go as far as to specify that objects referenced from within the tree to content that is not current evidence must be listed in the pertinent other evidence sequence (though that was the intent).

The attribute is called “pertinent other” rather than “prior” evidence sequence, since not all relevant information is necessarily acquired prior to the current procedure. For example, two more or less concurrent but separately requested procedures might be cross-referenced in each others’ reports.<sup>25</sup>

---

25. This is not the same as the case where a single report satisfies two requests, and it is copied into both studies then cross-referenced via the Identical Documents Sequence, as has already been described.

## Request and Procedure Information

In the world of managed workflow, there is a clear distinction between what is requested and what is performed. In many cases these will be the same, but not always. In either case, it is important to convey information about both in composite object instances.

In the case of a report, there may be an implicit association of the report with the procedure that the report is about. Or, there may be a specific (separate) request to create a report. In the former case, the report will be part of the same study as the procedure itself. In the latter case, it will be in a different study. If a single report satisfies multiple requests, and is replicated as an “identical document” in the corresponding studies, then each instance of the identical document should contain information about both requests.

Accordingly, in the SR Document General Module there is a Referenced Request Sequence (0040,A370) attribute which is required to be present if a request exists. It contains one or more items with:

- ▶ a Study Instance UID (0020,000D), which would normally be obtained from a modality worklist response, and used in generated composite objects
- ▶ a Referenced Study Sequence (0008,1110), which may contain a single “alias” of the Study Instance UID
- ▶ an Accession Number (0008,0050), which is usually a department-specific order number
- ▶ Placer (0040,2016) and Filler (0040,2017) Order Numbers for the Imaging Service Request, from an HL7 interface to an information system
- ▶ a Requested Procedure ID (0040,1001), which usually gets mapped into Study ID (0020,0010) in generated composite objects
- ▶ a Requested Procedure Description (0032,1060), which usually gets mapped into Study Description (0008,1030) in generated composite objects
- ▶ a Requested Procedure Code Sequence (0032,1064), which usually gets mapped into Procedure Code Sequence (0008,1032) in generated composite objects

On the performed side, contained within the SR Document General Module is:

- ▶ a Performed Procedure Code Sequence (0040,A372), which may contain zero or more codes “pertaining to this SOP Instance”<sup>26</sup>

Since all structured reports are composite objects, they contain the General Study Module, which contains various identifying and descriptive attributes of the study that the report is part of. These include:

---

26. Notice that the presence of this attribute is probably a mistake, since there is already in the General Study Module, an optional attribute Procedure Code Sequence (0008,1032) which is defined to mean “the type of procedure performed,” and may contain one or more items. It probably would have been more appropriate in the SR Document General Module to simply specialize the type of the existing attribute rather than create a new one. The best practice may therefore be to put the same codes in both attributes, to be sure that a receiver will find one or the other.

- ▶ Study Instance UID (0020,000D)
- ▶ Referenced Study Sequence (0008,1110), which may contain zero or more “aliases” of the Study Instance UID
- ▶ Accession Number (0008,0050)
- ▶ Study ID (0020,0010)
- ▶ Study Description (0008,1030)
- ▶ Procedure Code Sequence (0008,1032)

In the degenerate case, one report is “implicitly” performed for one procedure in response to one request, and the procedure that is performed is the same procedure as that which was requested. If so, then the following values should be present and equal in all of:

- ▶ a single item of the Referenced Request Sequence in the report
- ▶ the General Study Module in the report
- ▶ the General Study Module in any other composite objects from the same procedure (such as images and waveforms)

The identical or corresponding attributes are:

- ▶ Study Instance UID
- ▶ Accession Number
- ▶ Requested Procedure ID and Study ID
- ▶ Requested Procedure Description and Study Description
- ▶ Requested Procedure Code Sequence, Performed Procedure Code Sequence and Procedure Code Sequence

Referenced Study Sequence should be empty or absent everywhere.

## SR Document General Module

Summarizing the preceding sections, the SR Document General Module contains:

- ▶ indication of whether the document is “partial” or “complete”
- ▶ indication of whether the document is “verified” or “unverified”
- ▶ if verified, who verified the document and when
- ▶ when the content of the document was created
- ▶ lists of predecessor and identical documents
- ▶ lists of current and prior evidence
- ▶ information about the request and the procedure performed

## Example of Management Information Encoding

The following is an example of how to encode the management information of a simple report that is part of a study which also contains the images that satisfy the requested procedure. In this example, what was performed is the same as what was

requested, no HL7 integration is available, and performed procedure steps are not being generated.

For completeness, the contents of the SR Document Series Module have also been included, as have been some optional attributes of the SOP Common Module that are of relevance, such as Specific Character Set, Instance Creation Date and Time, Instance Creator UID and Timezone Offset From UTC.<sup>27</sup> The SOP Class and SOP Instance UIDs are mandatory attributes of the SOP Common Module, and are always included.

```
(0008,0005) Specific Character Set "ISO_IR 100"
(0008,0012) Instance Creation Date "20000701"
(0008,0013) Instance Creation Time "163215"
(0008,0014) Instance Creator UID "0.0.0.0.1"
(0008,0016) SOP Class UID "1.2.840.10008.5.1.4.1.1.88.11"
(0008,0018) SOP Instance UID "0.0.0.0.1234.2.1"
(0008,0023) Content Date "20000701"
(0008,0033) Content Time "163210"
(0008,0050) Accession Number "AN20000701"
(0008,0060) Modality "SR"
(0008,0201) Timezone Offset From UTC "-0500"
(0008,1030) Study Description "Wrist"
(0008,1032) Procedure Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "88.23"
(0008,0102) Coding Scheme Designator "I9C"
(0008,0104) Code Meaning "Skeletal x-ray of wrist and hand"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0008,1110) Referenced Study Sequence
(fffe,e000) Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0008,1111) Referenced Study Component Sequence
(fffe,e000) Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0020,000d) Study Instance UID "0.0.0.0.1234"
(0020,0010) Study ID "9526"
(0020,000e) Series Instance UID "0.0.0.0.1234.2"
(0020,0011) Series Number "2"
(0020,0013) Instance Number "1"

(0040,a370) Referenced Request Sequence
(fffe,e000) Item
(0008,0050) Accession Number "AN20000701"
(0008,1110) Referenced Study Sequence
```

27. The contents of the General Patient, Patient Study, Specimen Identification and General Equipment Modules have not been included, since these are discussed in the chapter on "Context: Who did what to whom, when..." under the subject of initial (default) context.

```

(fffe,e000) Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0020,000d) Study Instance UID "0.0.0.0.1234"
(0032,1060) Requested Procedure Description "Wrist"
(0032,1064) Requested Procedure Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "88.23"
(0008,0102) Coding Scheme Designator "I9C"
(0008,0104) Code Meaning "Skeletal x-ray of wrist and hand"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,1001) Requested Procedure ID "9526"
(0040,2016) Placer Order Number/Imaging Service Request ""
(0040,2017) Filler Order Number/Imaging Service Request ""
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

(0040,a372) Performed Procedure Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "88.23"
(0008,0102) Coding Scheme Designator "I9C"
(0008,0104) Code Meaning "Skeletal x-ray of wrist and hand"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

(0040,a073) Verifying Observer Sequence
(fffe,e000) Item
(0040,a075) Verifying Observer Name "Clunie^David^A^Dr."
(0040,a088) Verifying Observer Identification Code Seq.
(fffe,e000) Item
(0008,0100) Code Value "1234567"
(0008,0102) Coding Scheme Designator "99BIGBROTHER"
(0008,0104) Code Meaning "Clunie^David^A^Dr."
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(0040,a027) Verifying Organization "PixelMed Publishing"
(0040,a030) Verification DateTime "20000701163214-0500"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

(0040,a375) Current Requested Procedure Evidence Sequence
(fffe,e000) Item
(0008,1115) Referenced Series Sequence
(fffe,e000) Item
(0008,1199) Referenced SOP Sequence
(fffe,e000) Item
(0008,1150) Referenced SOP Class UID
                    "1.2.840.10008.5.1.4.1.1.1.1"
(0008,1155) Referenced SOP Instance UID "0.0.0.0.1234.1.1"
(fffe,e00d) Item Delimitation Item

```

(ffff,e0dd) Sequence Delimitation Item  
(ffff,e000) Item  
(0008,1150) Referenced SOP Class UID  
  "1.2.840.10008.5.1.4.1.1.1.1"  
(0008,1155) Referenced SOP Instance UID "0.0.0.0.1234.1.2"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0020,000e) Series Instance UID "0.0.0.0.1234.1"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0020,000d) Study Instance UID "0.0.0.0.1234"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item

(0040,a385) Pertinent Other Evidence Sequence  
(ffff,e000) Item  
(0008,1115) Referenced Series Sequence  
(ffff,e000) Item  
(0008,1199) Referenced SOP Sequence  
(ffff,e000) Item  
(0008,1150) Referenced SOP Class UID  
  "1.2.840.10008.5.1.4.1.1.1"  
(0008,1155) Referenced SOP Instance UID "0.0.0.0.1000.1.1"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(ffff,e000) Item  
(0008,1150) Referenced SOP Class UID  
  "1.2.840.10008.5.1.4.1.1.1"  
(0008,1155) Referenced SOP Instance UID "0.0.0.0.1000.1.2"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0020,000e) Series Instance UID "0.0.0.0.1000.1"  
(ffff,e00d) Item Delimitation Item  
(ffff,e000) Item  
(0008,1199) Referenced SOP Sequence  
(ffff,e000) Item  
(0008,1150) Referenced SOP Class UID  
  "1.2.840.10008.5.1.4.1.1.88.11"  
(0008,1155) Referenced SOP Instance UID "0.0.0.0.1000.2.1"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0020,000e) Series Instance UID "0.0.0.0.1000.2"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item  
(0020,000d) Study Instance UID "0.0.0.0.1000"  
(ffff,e00d) Item Delimitation Item  
(ffff,e0dd) Sequence Delimitation Item

(0040,a491) Completion Flag "COMPLETE"  
(0040,a492) Completion Flag Description "For release"

(0040,a493) Verification Flag “VERIFIED”

Notice in this example that:

- ▶ there is no Predecessor Documents Sequence since there was no previous version of the document created
- ▶ there is no identical Documents Sequence since there are no identical documents, i.e. there is only one report for one procedure for one request
- ▶ Retrieve AE Title and associated attributes are not used in the Current Requested Procedure Evidence Sequence
- ▶ the two Digital X-Ray images that are the entire output of the diagnostic procedure requested are listed in Current Requested Procedure Evidence Sequence
- ▶ the images are in the same study as the report, but in a different series (as is apparent from the different Series Instance UIDs)
- ▶ the Pertinent Other Evidence Sequence not only lists both the Computed Radiography images from the prior study, but also the Basic Text Structured Report of the previous study, which is of course in a different series to the images
- ▶ the attributes in the SR Document General Module that describe the request, what was performed and what is in the General Study Module are consistent, as previously described
- ▶ from within the SR tree, one can refer to instances of current or pertinent other evidence by their SOP Instance UID, and search the information from the sequences in the SR Document General Module to find the corresponding Series and Study Instance UIDs

## SR Document Series Module

In the previous example, the SR Document Series Module was mentioned. The Series entity has very little meaning for SR documents, but like all composite DICOM objects, the series entity has to be modeled and encoded. There are no semantics specified for an SR series beyond the basic DICOM series requirements that:

- ▶ all objects in a series be generated by the same equipment
- ▶ all objects in a series have the same modality

Specifically, these rules mean that reports and measurements encoded in SR documents cannot be in the same series as the images or waveforms to which they refer.

This module contains:

- ▶ Modality (0008,0060), a mandatory attribute which is always set to “SR”
- ▶ Series Instance UID (0020,000E)
- ▶ Series Number (0020,0011)
- ▶ Referenced Study Component Sequence (0008,1111)

It is bad form to always set the Series Number to “1” for all generated structured reports, since this number will almost certainly be used by one of the series of images. Although two series can be separated by their Series Instance UID (which *must* always

be different), it irritates users intensely to see different series with the same number. Since a reporting application will very likely have access to the entire rest of the study, there is seldom any excuse other than laziness for not assigning a meaningful series number.<sup>28</sup>

The Referenced Study Component Sequence may be empty, but if not, it contains the Referenced SOP Class and Instance UIDs of a Performed Procedure Step. Though it is beyond the scope of this book to discuss the details of managing modality scheduled and performed procedure steps, it suffices to say that:

- ▶ procedures are scheduled and performed as a series of “procedure steps”
- ▶ acquisition devices receive a list of “Scheduled Procedure Steps” (SPS) through the “Modality Worklist” (MWL) service
- ▶ acquisition devices may report the progress and completion of “Performed Procedure Steps” (PPS) through the “Modality Performed Procedure Step” (MPPS) service, as well as record identifiers of the step in Referenced Study Component Sequence
- ▶ theoretically, generation of a report may be part of the same PPS as the acquisition of its evidence, but very often it will be a separate step
- ▶ at the present time, there are no explicit reporting or interpretation SPS or PPS entities or services defined<sup>29</sup>

For the time being it is important to know that the value, if any, that should be stored in Referenced Study Component Sequence in a report document, is that UID corresponding to the interpretation PPS, not the evidence acquisition PPS, unless they are the same. In other words, this attribute references

- ▶ nothing if there is no PPS at all for the interpretation
- ▶ an interpretation PPS, once such an entity is defined in the standard
- ▶ a modality PPS, if the report is generated as part of the acquisition of the evidence

In the case where the interpretation PPS is not part of the acquisition, the evidence will always be accessible in the usual manner through the Current Requested Procedure Evidence Sequence of the SR Document General Module.

## Query and Retrieval

Up to this point, how individual document objects are constructed has been described, in addition to how they may reference each other in terms of prior versions, copies of identical reports, as well as how they reference other objects that are the evidence on which the document is based. Like other DICOM composite objects (such as images and waveforms), structured reports reside in a repository or archive.

---

*28. It should be obvious that document instances in Instance Number shouldn't always be a constant value like "1" either, unless they really are the one and only instance in a report series, as is often the case.*

*29. Although this situation will soon be rectified by the adoption of Supplement 52, Interpretation Worklist.*

Such archives store objects that are sent to it by applications that create structured reports.

How then, do users of reports get access to them? Two approaches are possible using DICOM services:

- ▶ reports are “pushed” to wherever it is anticipated that they might be needed (using the Storage service class)
- ▶ reports are “pulled” on demand (using the Query-Retrieve service class)

It might sound ridiculous to “push” reports everywhere, but such an approach may be expedient in many simple cases with few outlets on which to view reports. It is analogous to “snail mailing” reports directly to their intended recipients. In many cases the consumer of the report may actually be a gateway to an information system which uses a broadcast message rather than document paradigm.<sup>30</sup> In the latter case, the “push” mechanism is often used to distribute the report to other archives, databases and servers, which will then provide access to the report content through some other means.<sup>31</sup>

The existing query-retrieve mechanism is widely supported by DICOM image archives and workstations. It is only a small step to extend such devices to also store, search for and retrieve composite objects other than images. This is one of the primary reasons that it was decided to encode structured reports in DICOM as composite objects rather than extending the old normalized reporting services.<sup>32</sup>

The query-retrieve service class allows a user to:

- ▶ search for individual studies, series and instances based on “matching keys” such as identifiers and names
- ▶ additionally ask for more information about each matched instance by specifying “return keys”
- ▶ request that studies, series and instances that are matched be “retrieved” and transmitted, either to the originator of the retrieval request or to another device

Performing the query and performing the retrieval are actually two different operations (though they share the same mechanisms for specifying keys and responses). Accordingly, there are different sets of SOP classes. The C-FIND SOP classes perform queries and return information about matches as responses. The C-MOVE SOP classes initiate retrieval and return status responses.<sup>33</sup>

These services are used to request a list of reports or images that match certain criteria (e.g. all studies from today), and to populate a “browser” on the screen that lists all the responses. Then some subset of the listed responses is selected, often a single

---

30. Such as HL7 V2.x ORU messages with reports in OBX segments. See also the chapter on “Other Standards: HL7, MIME, XML, COAS.”

31. Such as a proprietary application or a web server.

32. The detached report and interpretation management services have never been popular or widely supported, and may well be “retired” from the standard at some time in the future. Structured reporting is completely independent of these older services.

patient, study or report, and a retrieval request containing their unique identifiers (UIDs) is sent<sup>34</sup>. This two step operation is performed using a DICOM C-FIND to return a list of responses, from which UIDs are selected, and initiating one or more DICOM C-MOVE operations using the UIDs.

Notice that it is possible to perform the retrieval at the study or series level, not just at the individual instance level. Indeed one can retrieve lists of studies, series and instances in a single operation.<sup>35</sup>

Care should be taken when retrieving entire series or studies that the storage SOP classes of the contained objects are actually supported by the receiving device. For example, a study may contain objects of various image, waveform and report storage SOP classes. If a workstation that performs a retrieval of a study doesn't support all of the necessary storage SOP classes, then some of the transfers will fail. Hopefully, the SCP will successfully transmit all the instances of storage SOP classes that are supported,<sup>36</sup> but this is not guaranteed. The responses returned over the C-MOVE association will indicate which transfers have succeeded and which have not.

## Query Information Models

The most widely supported query model in DICOM is the so-called "Study Root" model.<sup>37</sup> In this model, the highest entity is considered to be the study, and attributes of any higher level entities, such as the patient are considered to be attributes of the study. In other words, as far as the query is concerned, there is no such thing as a patient per se, only studies, each of which contains attributes that describe a patient. In this model, below the study is the series and then the instance level.

This distinction would be of little consequence were it not for the fact that most current systems only support the "hierarchical" query mechanism, the minimum required by the standard. What this means is that:

- 
33. *The actual transmission of objects initiated by the C-FIND occurs using the storage service class, over one or more separate associations than that used for the C-FIND. There is also a C-GET set of SOP classes that retrieve objects over the same association. This is largely unsupported and hence should be avoided.*
  34. *It is not possible to retrieve with keys other than UIDs, e.g. by matching on the basis of name or other identifier. This is fortunate, since a poorly formulated query that also retrieved could potentially generate an enormous volume of transfers. In other words, one has to determine precisely what to retrieve first.*
  35. *The standard was a little confusing as to whether "list of UID matching" was permitted above the instance level. A recent correction proposal (CP 146) has clarified that indeed lists of studies and series as well as instances can be specified in a C-MOVE.*
  36. *Supported in the sense that Presentation Contexts (SOP class plus transfer syntax) can be negotiated when the C-STORE association is established.*
  37. *There is also a "Patient Root" model and a "Patient/Study Only" model, but these are so rarely supported that they aren't even worth considering. Hopefully someday they will be retired from the standard. Just be aware that if one device supports only one model, and the other device supports only a different model, they will not be able to interoperate.*

- ▶ for a C-FIND (a query), one must uniquely identify all the higher level entities before matching attributes of a lower level entity
- ▶ for a C-MOVE (a retrieval) one must uniquely identify all the higher level entities before retrieving a lower level entity

Some examples will clarify this.

Suppose one wanted to search for all the studies performed on a particular date. No problem, just do a C-FIND at the “STUDY” level specifying a matching key of Study Date. This is fine since the study level is the top level.

However, suppose that one wanted to find all images with a modality of “CT” performed on a particular date. The Modality attribute is defined at the series level in DICOM, so this query involves matching keys from two levels. One cannot just send a “SERIES” level query with Study Date and Modality as matching keys. The rules of the hierarchical query specify that a single value of the unique key of the higher level has to be specified whenever matching on a lower level. In this example, one would have to send a single Study Instance UID value and could use only Modality as a matching key.

In other words, to perform the specified search as described above involves several steps:

- ▶ first, do a study level query to find all studies on a particular date
- ▶ then, for each of the studies found, perform another query at the series level for a particular modality

Notice that this doesn't need only two successive queries, it needs  $1+2*n$  queries, where  $n$  is the number of responses generated by the first query.

If this sounds like a lot of work, it is. There are two solutions. One is a quick and dirty pragmatic solution which is occasionally applied (by the authors of the standard) for really important lower level attributes. That solution is to redefine multi-valued equivalents at a higher level in the model. For example, the case of querying for modality is so common that a new, optional, “Modalities in Study” attribute has been added at the study level.<sup>38</sup> Changes to the standard like this are very rarely made, and this is not a general solution.

A better approach is simply to support “relational queries.” These have been defined in DICOM since the start, but have always been optional. Relational queries simply waive the requirement that unique keys have to be provided for higher levels than the level of the query. In other words, in the previous example one could, in a single query, request matching on both a study and a series level attribute at the same time, without having to specify a unique Study Instance UID.<sup>39</sup>

---

*38. Unfortunately, since it was added as a correction proposal (CP 71), is relatively new, not everyone agrees it is a good idea, and it requires either changing the database schema or doing a relational query inside the SCP anyway, not all SCPs support “Modalities in Study”. This means all SCUs have to be prepared to work properly without it.*

The same considerations apply to retrieval. In the hierarchical approach, one cannot just retrieve, for example, instances by their SOP Instance UID alone. One also must specify the unique keys of the higher entities. Specifically, one must send the Study Instance UID and Series Instance UID in the request as well.<sup>40</sup> If the relational mechanism is supported, then one can retrieve objects by SOP Instance UID alone.

The introduction of structured reporting exacerbates the query problem. Not only is content more complex, in the sense that there are more types of objects and new query keys to deal with, but references to other objects are of greater importance. Historically, even though most modern devices are relational “inside,” based as most are on commercial-off-the-shelf databases,<sup>41</sup> only the hierarchical DICOM mechanism has been implemented. It is expected that this will change and that the relational model will be more widely implemented to support SR.

### **Query Keys: Matching and Return**

The C-FIND service is based on the notion that the SCU sends a request “identifier” that contains “matching” and “return” keys. The SCP returns a response for each object at the query level that matches the request identifier. The identifiers in both the request and the response consist of an ordinary DICOM data set of attributes.

In the request, an attribute may have either a zero length value, in which case it is essentially a request to “return” the specified attribute in the responses.<sup>42</sup> If an attribute contains a value, then “matching” really is performed. Or at least it may be, if the SCP supports that attribute as a key. Only a few attributes at each level of the query information model are “required” to be supported by the SCP. In the case of the Study Root model, the following keys are required at the Study level:

- ▶ Study Date, Study Time, Study ID and Accession Number
- ▶ Patient’s Name and Patient ID
- ▶ Study Instance UID (which is the unique key at this level)

At the Series level, the following keys are required:

- ▶ Modality

---

39. It is still necessary in the relational query to specify a query “level,” below which matching will not be performed.

40. In structured reports, even though references to objects made from within the tree itself are by SOP Instance UID alone, a repository of corresponding Study and Series Instance UIDs is provided in the general modules. This considerably simplifies retrieving them using the hierarchical mechanism. In older DICOM objects, references from within a composite object to other objects are usually only by SOP Instance UID. This makes the referenced objects very hard to find if they are not part of the same series, much less the same study.

41. This has not always been the case. The simple hierarchical study root model gained prominence as a consequence of the relatively crude way in which early CT and MR scanners and workstations modeled information internally, often without the benefit of any real database at all.

42. The way this is described in the standard is that “universal matching” is performed for those attributes with zero length, i.e. they match anything.

- ▶ Series Number
- ▶ Series Instance UID (which is the unique key at this level)

At the Instance level,<sup>43</sup> the following keys are required:

- ▶ Instance Number<sup>44</sup>
- ▶ SOP Instance UID (which is the unique key at this level)

A “unique key,” by the way, is the attribute that uniquely identifies an object at the specified level of the query. That is, there may not be two instances with the same SOP Instance UID; any other instance level attribute may be the same for two objects, except the unique key. There is always one and only one attribute specified as a unique key for each level, and it is always a UID<sup>45</sup>.

As an SCU, one can depend on being able to return or match on required or unique keys. In the definition of the query service, the standard also specifically lists a number of optional attributes at each level which an SCP may choose to support or not. In addition, for the Series and Instance levels (only<sup>46</sup>) it specifies that “all other attributes” at the corresponding level may also be specified as optional keys.

In the real world, many C-FIND SCPs are quite generous in the additional optional attributes that they support. These typically include various dates, times, numbers, identifiers, codes and descriptions. Precisely which optional attributes are supported will be specified in the device’s conformance statement. It is likely to be some time, however, before these same SCPs will add optional attributes specific to structured reporting, as will be discussed later.

While on the subject of optional keys, it is entirely at the discretion of the SCP as to whether an optional key will be:

- ▶ ignored entirely, in which cases it will neither be matched or returned
- ▶ returned, but not matched
- ▶ matched and returned

---

43. Which is equivalent to the level of the image, waveform or report.

44. Instance Number was originally referred to as Image Number, but in order to generalize the query mechanism to apply to all composite instances, not just images, it was renamed, and is used in all new composite objects, including waveforms and reports.

45. With the one exception of the patient level in the Patient root model. Patients don’t generally come with unique identifiers, and so Patient ID is used instead. This is a problem, because such things as “medical record numbers” that are used for Patient ID are often unique within an institution, but not beyond. In practice this is less of a problem for DICOM queries, since the Patient Root model is rarely implemented.

46. Why this is not also the case for the study level is not immediately obvious. Optional attributes (like Modalities in Study) have subsequently been added to the optional attribute list for the study level in correction proposals. Accordingly, it is probably fairly harmless to request any other attribute at the study level also, and to support other attributes as an SCP. If challenged, one can argue that a standard extended SOP class is being used.

There is no way for the SCU to know until it tries. For this reason, if the SCU wishes to select out only the responses that match an optional key, if the SCP at least returns the key, the SCU should “filter” the responses itself, just in case the SCP did not match on the key.

Matching has been discussed, without being too specific about what matching criteria are permitted. If the SCP matches on a key, it must support:

- ▶ single value matching, i.e. an exact match on the complete value specified
- ▶ list of UID matching, where a match on one of a number of UIDs is specified
- ▶ universal matching, where a zero-length key matches any value (i.e. this is how one specifies a return key, as already described)
- ▶ wild card matching, where ‘?’ and ‘\*’ characters are used in the usual manner to match any character or characters amongst specified values
- ▶ range matching for dates and times, where either a start or an end or both may be specified
- ▶ sequence matching, where a single item containing multiple attributes may match any one of multiple items in a sequence attribute

Note that all matching is case sensitive, except for attributes with a Person Name (PN) value representation.<sup>47</sup> Notice also that many of the special characters for specifying matching in a key are outside the usual repertoire for the value representation of that attribute. This is explicitly permitted.

Also notice that when matching an attribute with multiple values, if any one of the values matches the key, then the match is considered successful, and all the values of the attribute are returned in the response. It is not possible to specify multiple values in the matching key however.<sup>48</sup>

## Query Keys for SR

Structured reporting poses some interesting challenges for the existing query-retrieve mechanism. This is particularly true when specifying what keys to match on while searching, and what keys to return containing values.

Most of the “top level” administrative attributes in the general modules are not a problem, since SCPs will support the usual patient, study, series and instance identifying attributes just as they do for images. In the case of SR however, it is also desirable to extend this list to include:

---

47. This has not always been the case, but has been allowed as a result of a recent correction (CP 196). In reality, many implementations in the past ignored the letter of the law, and performed case insensitive matching of names anyway.

48. For example, if matching the Image Type attribute with the value of “ORIGINAL\PRIMARY”, then a key of either “ORIGINAL” or “PRIMARY” would match and return “ORIGINAL\PRIMARY”. Likewise, a key of ORIGINAL would return “ORIGINAL\PRIMARY” or “ORIGINAL\SECONDARY”. It is not possible to distinguish the two by matching. That is, one cannot send “ORIGINAL\PRIMARY” as a key.

- ▶ management attributes of the SR Document General Module, particularly those pertaining to the completion and verification status, the person, date and time of verification, as well as the lists of current and pertinent other instances
- ▶ attributes of the root content item, particularly the document title that is stored in the top-level Concept Name Code Sequence<sup>49</sup>

Largely this is a matter of SCPs being modified to include these attributes in their database, and to expose them as optional matching keys.

Since sequences are used extensively in SR, the return and matching of sequence attributes is important to consider. Recall that for any attribute, whether it be a sequence or not, if it is supported by the SCP as a required or optional key, a zero length key will specify universal matching, i.e. the value of the attribute will always be returned. If the attribute is a sequence like Verifying Observer Sequence or Current Requested Procedure Evidence Sequence, then the entire contents of the sequence will be returned in the response identifier. Likewise, if the specified key is Concept Name Code Sequence, then the document title will be returned as a coded entry.

Notice that if one specified the Content Sequence as a return key, if the SCP supported it, then the entire SR tree below the root node would be returned. It is probably not a very good idea to do this. However, there may be circumstances in which part of the Content Sequence is needed. For example, if the document title were post-coordinated, then the Concept Name Code Sequence might not be sufficiently precise. One could specify the Content Sequence as a matching key, not with universal matching (zero length), but as a single Item that contained a value of Relationship Type that was "HAS CONCEPT MOD". If the SCP supported matching on the Content Sequence, then only items that were children of the root node that were modifiers of the document title would be returned.<sup>50</sup>

Since so many SR constructs use coded entries, is it possible to match on sequence items within sequence items? The standard does explicitly call out that one is required to perform "recursive sequence matching," unless of course, one doesn't support matching on any sequence items at all. In theory, if one were to specify an item of a coded sequence attribute containing a code value and coding scheme designator<sup>51</sup>, the coded sequence attribute itself nested within an item of another (outer) sequence, then only those "items within items" would match. For example, given a report with a root node containing:

```
<CONTAINER:(209076,99PMP,"Chest X-ray Report")>  
  <contains CONTAINER:(209002,99PMP,"Findings")>
```

---

49. It is for this reason that the root node is "exposed" or folded into the top level data set, rather than nested as a single item in a content sequence. The latter would have been tidier, but much harder to match during a query.

50. Of course, if the SCP supported return of Content Sequence without matching, then the entire SR tree would be returned once again. Obviously, if an SCP is going to support this key, it would do well to support matching, not just return. Conversely, the SCU should be prepared to discard all items except those it asked to match on in the first place.

51. There would be no point in specifying code meaning.

```

...
...
<contains CONTAINER:(209018,99PMP,"Diagnoses")>
  <contains CODE:(209017,99PMP,"Diagnosis")=
    (197.0,I9C,"Secondary malignant neoplasm of lung">
...

```

then a query of the form:

```

(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a010) Relationship Type "CONTAINS"
(0040,a040) Value Type "CONTAINER"
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209018"
(0008,0102) Coding Scheme Designator "99PMP"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

```

should return:

```

<CONTAINER:(209076,99PMP,"Chest X-ray Report")>
  <contains CONTAINER:(209018,99PMP,"Diagnoses")>
    <contains CODE:(209017,99PMP,"Diagnosis")=
      (197.0,I9C,"Secondary malignant neoplasm of lung">

```

having matched the Relationship Type, the Value Type and the Concept Name Code Sequence of one of the children of the document root, and returned the entire subtree corresponding to the match (and nothing more).

Alternatively, if one knew that all diagnoses were stored as content items within a container child of the root node, but that the name of that container varied, one could just search for the concept name corresponding to "diagnosis" at that level, as follows:

```

(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a043) Concept Name Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "209017"
(0008,0102) Coding Scheme Designator "99PMP"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

```

This should return the same as the last query:

```

<CONTAINER:(209076,99PMP,"Chest X-ray Report")>
  <contains CONTAINER:(209018,99PMP,"Diagnoses")>
    <contains CODE:(209017,99PMP,"Diagnosis")=
      (197.0,I9C,"Secondary malignant neoplasm of lung">

```

but this time having matched the Concept Name Code Sequence of the child of the child of the document root. This query would return any diagnosis at this nesting level, whether it were a code value type or a text value type, since the matching keys are specified for the concept name only.

Alternatively, if one were looking for the specific coded diagnosis *at the same level of nesting*, one could send the query:

```

(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a730) Content Sequence
(fffe,e000) Item
(0040,a168) Concept Code Sequence
(fffe,e000) Item
(0008,0100) Code Value "197.0"
(0008,0102) Coding Scheme Designator "I9C"
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item
(fffe,e00d) Item Delimitation Item
(fffe,e0dd) Sequence Delimitation Item

```

This should again return the same result:

```

<CONTAINER:(209076,99PMP,"Chest X-ray Report")>
  <contains CONTAINER:(209018,99PMP,"Diagnoses")>
    <contains CODE:(209017,99PMP,"Diagnosis")=
      (197.0,I9C,"Secondary malignant neoplasm of lung">

```

but this time having matched the Concept Code Sequence, i.e. the value rather than the concept, of the child of the child of the document root.

The point of the foregoing examples is to illustrate that the DICOM query mechanism only allows one to query within sequences when one knows the depth of nesting at which the match will occur. This is a major limitation, since in general the location (depth of nesting) of an item of interest is not always predictable except in very simple structures.

The existing DICOM mechanism essentially allows one to specify a "pattern" that is a sub-tree that is to be matched against the real tree. The problem is that this pattern must always be anchored at the root. A task for designers of a future query service for DICOM SR would be to allow specification of a pattern:

- ▶ that could match anywhere in the tree, not just from the root
- ▶ that was not restricted to contiguous descendants, i.e. there could be "gaps" in the pattern that matched any depth of nesting

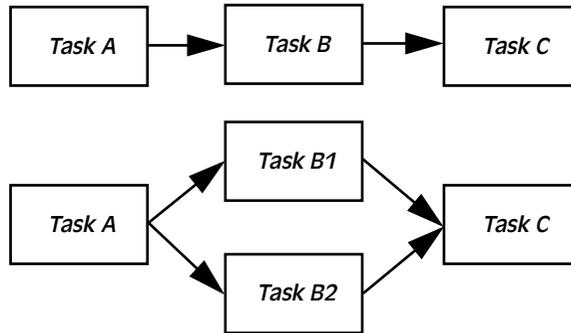


FIGURE 43. Workflow Management Tasks

There are no doubt other features that would also need to be considered to be useful.<sup>52</sup>

Another idea that has been discussed (in the context of better query models for structured reporting) is the notion of the “fuzzy match.” Loosely translated, this means “give me what I need, not exactly what I asked for.” The presumption is that the SCP is often in a better position to judge how best to select a response to a query, rather than performing slavishly with literal matches. In the case of persons’ names for example, an SCP could experiment with permutations of ordering, spelling and pronunciation, using its knowledge of phonetic equivalents perhaps,<sup>53</sup> to choose a better match than a literal match. After all, the SCP is likely to know how it stores names in a “canonical” internal form, something the SCU may not be aware of.

## Workflow Management

Traditionally, DICOM systems have consisted of components, applications and devices that are “unmanaged,” in the sense that they perform their functions autonomously. For example, acquisition devices like CT scanners would scan patients as they arrived, sending their images to the archive either automatically or on request. Users would query the archive from workstations and retrieve those images they were interested in, and so on. In some cases, largely for performance reasons, images might be “pushed” to workstations in advance of their anticipated use. As systems become more complex, and composite objects develop inter-relationships (such as between images, waveforms, presentation states and reports), a better approach is required.

52. A good model to work from is the XPath specification that forms the basis of the “matching” that selects which rules in an XSL-T transformation pattern (style sheet) are applied to which parts of an XML document.

53. For example, by using the “Soundex” code, which is an indexing system that translates names into a 4 digit code consisting of 1 letter and 3 numbers. The US Bureau of the Census uses Soundex codes to index individuals listed in the census records. The advantage of Soundex is its ability to group names by sound rather than exact spelling. Genealogists are big on Soundex codes.

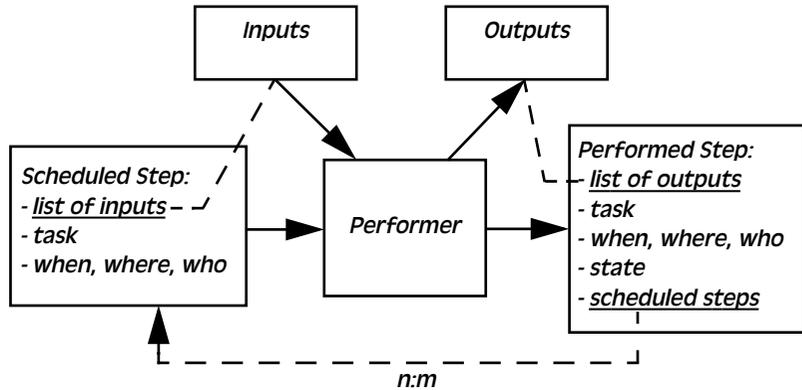


FIGURE 44. Scheduled and Performed Procedure Steps

Any process in an organization can usually be modeled as a series of steps or tasks. Individual tasks have inputs and outputs. Tasks can be scheduled. Tasks have a “state,” that is they may be scheduled, or in progress or completed. Normally, one cannot begin working on a task whose inputs are not yet available. If those inputs are generated as the output of a preceding task in the process, then the task is “dependent” on its predecessor. This task-oriented workflow model is illustrated in Figure 43. Notice in these examples that some tasks may proceed simultaneously, and that a succeeding task may be dependent on the output of more than one preceding task.

In DICOM, tasks are referred to as “procedure steps.” Procedure steps may be one of two types:

- ▶ scheduled procedure steps
- ▶ performed procedure steps

A scheduled procedure step is one which defines what work is to be done. It may include specification of information about what the nature of the task is, what inputs are required, and when, where and by whom it is to be done.

A performed procedure step is one which defines what is being, or has been, done. It includes specification of the state of the step, such as whether the work has been initiated, is in progress, has been completed or has been aborted. When the step has been completed, it may include a list of the outputs of the step. It may also contain a reference to the scheduled step or steps that the performed step “satisfies.”

Figure 44 illustrates the generalized model behind DICOM scheduled and performed procedure steps. Notice in particular that, in the real world, a performed step may satisfy more than one scheduled step and more than one performed step may be required to satisfy a single scheduled step, hence the “n:m” relationship. For example, if there are two scheduled procedure steps, one for a CT of the chest and one for a CT of the abdomen, they might be satisfied by a single performed procedure step that is a CT of the chest and abdomen. This is because the anatomical regions overlap, the

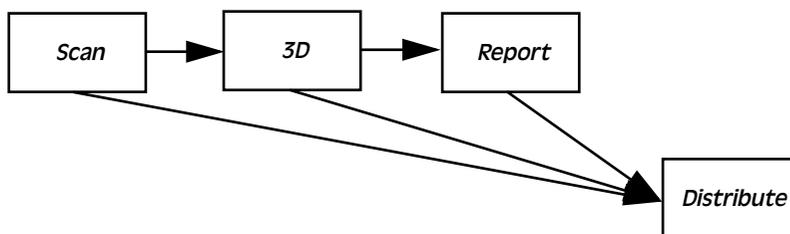


FIGURE 45. Dependencies of Reporting Procedure Steps

images may all be contained in the same series, and the same bolus of iodinated contrast agent may be used. Conversely, a single scheduled procedure step that is a CT angiogram of the Circle of Willis might require the images to be obtained on one device (the CT scanner) and the 3D reconstructions to be performed on another device, by definition (in DICOM anyway) a separate performed procedure step.<sup>54</sup>

In the context of structured reporting, it does not require a great intellectual leap to realize that the reporting process might well be managed as a series of workflow steps. In the example just illustrated, in addition to an acquisition step and a 3D reconstruction step, there could easily be a reporting step as well. In such a case, the reporting step could not be initiated until all the inputs were available, that is both the CT slices and the 3D reconstructions. Furthermore, the whole study should probably not be distributed to the referring physicians until the slices, the reprojections and the report are completed. Figure 45 illustrates this example.<sup>55</sup>

The reporting step is not necessarily a single step. In traditional radiology reporting where voice dictation is used, there may be “sub-steps” that involve such tasks as dictation, transcription, correction and verification, as illustrated in Figure 46.<sup>56</sup>

So far, DICOM has defined SOP classes to support workflow management at the acquisition device level, specifically:

► Modality Worklist

54. Ideally, the two steps of acquiring the images and performing the 3D reconstructions would be modeled as two separate scheduled steps. However, in practice there is a limit to the level of granularity of scheduling, beyond which efficiency may be impaired rather than improved.

55. In this example, the distribution task has been modeled as being dependent on all three earlier tasks, scanning, 3D reconstruction and reporting. Since the report creation is predicated on the availability of its preceding tasks, this may not be necessary. However, the availability of the images to the person performing the reporting task may not imply their availability for distribution to the referring user. Hence there may be additional work required. In the “bad old days,” this often used to involve retrieving the packet of films from the reporting radiologist and passing them along with the report to the requestor.

56. This is only an example. Another approach to voice dictated reporting might involve feeding back corrections made by the radiologist to the typist for re-dictation and subsequent re-verification. There might be additional steps for verification by a senior of a report dictated and corrected by a junior, and so on.

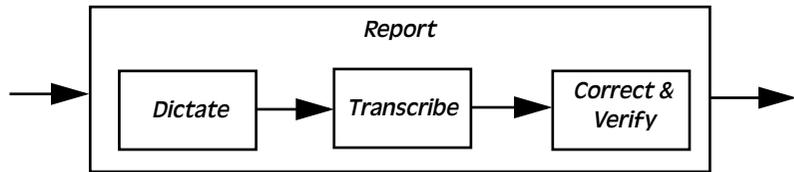


FIGURE 46. Reporting Step Decomposed into Smaller Steps

► Modality Performed Procedure Step

The Modality Worklist (MWL) support provides information to an acquisition device about what is to be done, where, when and by whom. In addition, it supplies identifying information about the patient that is the subject of the examination, as well as identification information for the study itself. This information is important for two reasons:

- patient identifying information can be made consistent throughout the system, rather than relying on error-prone and inefficient manual re-entry of the same data at the operator console
- study identifying information, specifically the Study Instance UID, can be generated by a single (central) source, to avoid later needing to coerce the UID in composite objects for a single study that were generated on different devices<sup>57</sup>

This information is supplied in the form of a list of “scheduled procedure steps.”<sup>58</sup> These are supplied in response to a query, very similar in form to a traditional composite object query.<sup>59</sup> What the DICOM MWL SOP class lacks with respect to the workflow management model discussed earlier is any state information. There is no instance of a normalized object that corresponds to a scheduled step. In particular, it is not possible for a modality that has decided to actually perform a scheduled procedure step to signal that it has done so (i.e. that the step is now in progress, or that it has been completed).<sup>60</sup>

The Modality Performed Procedure Step SOP class, on the other hand, does define instances of objects that are “performed procedure steps.”<sup>61</sup> These steps contain information about what was actually done, where, when and by whom, which may differ from what (if anything) was scheduled. The steps have a state which may be set to

57. That is, each modality that contributes objects to a study needs to use the same Study Instance UID, and this is very hard unless the Modality Worklist SOP Class is supported.

58. Or more precisely in the case of this SOP class, Modality Scheduled Procedure Steps (MSPS).

59. This design was used to allow reuse of components that implementers would already likely have constructed to support composite object queries.

60. Indeed it has been argued that the Modality Worklist is really a “view” of a database of information rather than a list of scheduled steps that have an existence in their own right.

61. Or more precisely in the case of this SOP class, Modality Performed Procedure Steps (MPPS).

in progress, completed or discontinued. The steps also contains lists of all the “outputs” of the step, in the form of lists of the composite objects generated. These lists are required to be sent by the time the state is set to complete.

Though each performed step does contain references to the scheduled steps that it has been performed in response to, notice that the correspondence is not necessarily 1:1. In particular, a single performed step may be performed to satisfy more than one scheduled step, or more than one step may be performed to satisfy a single scheduled step. It is beyond the scope of this chapter to go into further detail about modality specific workflow management. It suffices to say that this n:m relationship between scheduled and performed steps means that the setting of the state of a performed step cannot easily be used to “imply” a corresponding state change in a scheduled step.<sup>62</sup> In other words, it is difficult to know when to remove a scheduled step from a worklist.<sup>63</sup>

The modality worklist and modality performed procedure steps can be considered as DICOM’s first foray into the world of workflow management. To satisfy the requirement to manage workflow steps other than acquisition, new services and SOP classes are necessary. Up until now, these requirements have been satisfied in a proprietary manner. A very popular feature of some PACS is support for the concept of a “reporting worklist.” Such a worklist contains a list of studies that are ready for the radiologist to report. Some systems also support other worklists that list studies that are ready to correct and verify, and so on. However, being proprietary, such worklists can only be used with devices from the same vendor. If a user wants to make use of third-party image display and reporting workstations, then they usually have to sacrifice support for reporting worklists. While reporting worklists can be simulated by “pushing” all studies to be read to the workstation, or by querying for all studies performed by a specific radiologist on a specific date, such workarounds are not very reliable or accurate. Here is a clear boundary between devices or applications from different vendors that could benefit from standardization.

Accordingly, DICOM has begun work on a proposal for “interpretation worklists.”<sup>64</sup> The proposal consists of a matching pair of worklist and performed procedure step SOP classes, modeled after the corresponding modality specific SOP classes already in the standard. They differ from the existing SOP classes in that:

- ▶ the tasks to which they apply are generalized, rather than specific to acquisition; these are specified as coded entries

<sup>62</sup>. Particularly since a scheduled step has no persistence per se.

<sup>63</sup>. One heuristic that may be used is to remove a scheduled step when any performed step satisfying it has been set to a state of complete, after a certain interval of time. This allows for scheduled steps to go away quietly without preventing a second device from finding a modality worklist entry if it is necessary to perform another step. This approach fails to prevent another device from trying to repeat the original performed step as long as the scheduled step is still visible.

<sup>64</sup>. Interpretation Worklists are proposed in Supplement 52. The information given here is from the latest draft at the time of writing, the public comment draft dated “23 June 2000.”



- ▶ the scheduled step is persistent and has a state that may be managed; a device can indicate when it has begun to satisfy a scheduled step and when it has been completed or discontinued
- ▶ the scheduled step specifies a list of inputs (composite objects) that are necessary for the performance of the task
- ▶ the performed step has potential states that are generally applicable to any task, and not specific to the task of reporting or interpretation (in progress, completed or discontinued)

Notice that the new classes are named “interpretation” rather than “reporting”.<sup>65</sup> This is because they are expected to be useful for management of a broad range of workflow tasks, not just traditional radiology reporting. For example, there is no reason that an interpretation worklist could not be used to manage such tasks as mammography Computer Assisted Detection (CAD) by an automated device, or any other kind of image processing, such as 3D reconstruction. Indeed, given that coded entries are used to specify the task, and that the states for scheduled and performed steps are not task specific, almost any task could be managed. Some people have suggested that distribution of composite objects, or even printing of them, could be managed by such a generalized service. For example, the following work items are called out in the current draft of the proposal:

- ▶ image processing
- ▶ quality control (of acquired images)
- ▶ computer aided detection and/or diagnosis
- ▶ interpretation
- ▶ transcription
- ▶ report verification
- ▶ print

There is little point in going on to specify further details of the proposed services, since they will no doubt be different in the final standard. What can be said is that the tremendous gains in productivity and consistency that have been observed as a consequence of deploying modality worklist, will likely soon be available in a standard form for reporting as well. If at all possible, almost identical mechanisms will likely be used.

## **Architectural Issues - Integrating the Health-care Enterprise**

DICOM is a standard that applies to the boundaries between devices and applications. There has been a concerted effort to avoid defining a particular architecture, since that would imply a certain implementation, and might exclude certain vendors as well as stifle innovation. DICOM goes no further than to specify a shared information model so that devices can exchange standard messages meaningfully.

---

<sup>65</sup>. At least they are so named as of the time of writing. One faction would like to change the name to “workstation worklist.” No doubt by the time this book is in print, there will have been other suggestions.

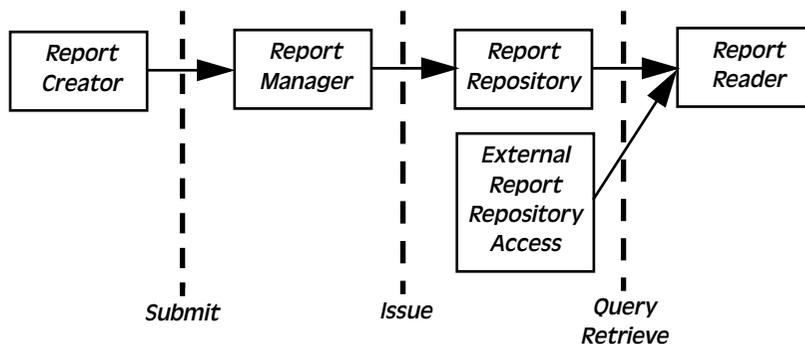


FIGURE 47. IHE Transactions and Actors Related to Reports

In order to build a working system, one needs more than just boundaries. One also needs components that interoperate across those boundaries, and together provide sufficient functionality to meet the user's requirements. The Integrating the Healthcare Enterprise (IHE) project<sup>66</sup> is an interesting exercise in prototyping such systems and demonstrating how standards like DICOM and HL7 can be used to actually construct systems.

IHE is mentioned here because it goes beyond the standards and specifies "actors" and "transactions" that such actors are required to support.<sup>67</sup> For example, in the case of an "acquisition modality" actor, the following transactions are specified:

- ▶ modality worklist provided
- ▶ modality procedure step in-progress/completed
- ▶ modality storage commitment
- ▶ modality images stored
- ▶ modality presentation state stored

One can see that IHE goes beyond defining conformance to individual SOP classes, but rather specifies a set of SOP classes that need to be supported in specific roles in order to fulfill the specified functionality of a particular actor.

In the case of reporting, the following actors are specified:

- ▶ report creator
- ▶ report manager
- ▶ report repository
- ▶ external report repository access

66. A joint initiative of the Radiological Society of North America (RSNA) and the Health-care Information and Management Systems Society (HIMSS).

67. The examples described here make use of requirements defined in the Year 2 Technical Framework document. See "<http://www.rsna.org/ihe/>" for further details.

- ▶ report reader

The following transactions are specified:

- ▶ report submission
- ▶ report issuing
- ▶ query report
- ▶ retrieve report

Figure 47, a subset of the systems transaction overview redrawn from the technical framework, illustrates how these actors and transactions fit together.

In IHE, the actors are fairly “atomic.” For example, in reality, a “report creator” actor might well be implemented together with an “image display” actor, in the case where a single device is used to both view images and report on them.<sup>68</sup>

The model envisaged is that:

- ▶ “report creators,” such as image review workstations or acquisition devices that generate measurements, will create draft DICOM structured reports and “submit” them (using a DICOM C-Store) to “report managers”
- ▶ “report managers” will take care of correcting, completing and verifying reports in some internal manner, and then “issue” verified DICOM structured reports (using a DICOM C-Store) to “report repositories”
- ▶ “report readers” will query “report repositories” for instances of DICOM structured reports (using a DICOM C-Find), retrieve them (using a DICOM C-Move and subsequent C-Stores) and then render them in some meaningful way for viewing

An interesting feature of the IHE approach is the specification of an “external report repository access.” This actor supports the same query and retrieval features as a report repository actor, but is intended to act as a “gateway” to external sources of information, such as those that provide laboratory or pathology information. The gateway is required to transcode information from external sources into DICOM Structured Reports for use “inside” the IHE technical framework.

Other features of IHE include constraints on the form and content of structured reports.<sup>69</sup> These are described as “use cases” and consist of:

- ▶ the key image note, a list of images and a text content item
- ▶ the simple image report, a one level list of headings, one level of text content items, and image references

---

*68. IHE deliberately avoids using conventional labels like “workstation” or “PACS” or “RIS” or “HIS” in order to avoid artificial divisions based on traditional vendor roles and responsibilities. There is no reason, for example, why a traditional IS vendor cannot support the “image archive” actor, or a traditional PACS vendor cannot support the “department system database” actor.*

*69. These constraints are specified in the form of “patterns,” the term “template” not really being applicable until DICOM Supplement 53 is completed.*

- ▶ the simple image and numeric report, which adds numeric content items to the simple image report

The rationale for these constraints is to keep the requirements simple for the first year in which IHE incorporates support for structured reports. The constraints do not mean that a report creator or manager cannot submit or issue a more complex report. They also do not mean that a report reader can violate the standard by failing to render unambiguously any report conforming to the SOP class that exceeds the simple use cases. They are only intended to provide a framework in which creators and managers can produce reports that they can expect readers to do a good job of rendering. Notice also that the current technical framework only requires support for the basic text SOP class, with the enhanced SOP class being optional. The comprehensive SOP class is not supported.<sup>70</sup>

The presence of a discussion of IHE here should not be taken to imply an endorsement that the IHE architecture is somehow the “preferred architecture,” nor that one can claim “compliance” to the IHE technical framework.<sup>71</sup> Rather it is intended as a non-partisan example of one way in which DICOM structured reporting services might be used to implement a working system.

## Architecture - Beyond DICOM

The IHE approach is in many ways traditional and limited, though practical and pragmatic. Though it avoids mention of traditional roles like “RIS” and “HIS” in favor of more generic actors, the workflow is conventional. In particular, discrete document objects move from one actor to another until they are finally made available in a repository for retrieval. In future years, as new DICOM services like interpretation worklist are provided, the management of individual report objects will no doubt become more sophisticated. As standard DICOM security services become available (as described in the chapter on “Security: Privacy and Authenticity”), these will no doubt be used to provide confidentiality, authentication and to restrict unauthorized access to reports.

Fundamentally however, there are limits to how sophisticated a system constructed using composite objects can be. While DICOM provides excellent support for the exchange of transient or persistent “documents” across the boundaries between devices, it is not, and does not claim to be, a distributed object management architecture. In particular, it fails to provide support for:

- ▶ distributed consistency between identical instances of the same object

---

70. In particular this means that “by-reference” relationships cannot be used in IHE year 2.

71. IHE goes out of its way to state that it is not creating a new standard, but rather illustrating how existing standards may be used to build systems. To quote the year 2 technical framework: “IHE is therefore an implementation framework, not a standard. Referencing IHE as a standard and claiming conformance to IHE are both inappropriate. Conformance claims shall be made in direct reference to specific standards. DICOM or HL7 conformance statements may, however, state that the products they describe are “implemented in accordance with the IHE Technical Framework.”

► reliable distributed transactions

For example, if it is necessary to fix an incorrect patient name at one location, it is not possible to specify that such a change be propagated across all objects in the system that reference that patient. Even if it were, there is no “two phase commitment” mechanism such that a transaction to multiple targets can be “rolled back” if it fails to succeed on all targets.

These examples of limitations are mentioned not to discredit DICOM, but rather to highlight the fact that constructing a completely functional system requires much more than cobbling together discrete components that are ostensibly compatible.

The architectures of existing PACS provide a good illustration of this problem. All PACS are surrounded by a boundary across which DICOM image objects are allowed to pass, inwards at the very least. Most provide an ability for external workstations to query and retrieve DICOM images, but often the use of third-party workstations in such a role will result in less functionality than proprietary workstations “inside” the PACS.<sup>72</sup> Many also provide modality worklists to acquisition devices, and accept performed procedure step notifications as well as perform storage commitment. All these services are on the boundary around the PACS which may be considered a monolithic device.

Some PACS make extensive use of DICOM internally, between components. This is especially true of commodity components like workstations and image archives that are often out-sourced. Depending on the network infrastructure, storage of image objects is often performed with a DICOM C-Store, sometimes with a proprietary compression transfer syntax to improve performance. Different PACS diverge substantially however, in terms of whether images are served from a central archive or pushed in advance to distributed nodes, and how the movement and management of image objects is performed. Almost all contain some sort of centralized database that contains the most up to date knowledge of where images are located, as well as any changes that have been made to names, studies that have been merged and so on. Most augment this centralized management with proprietary services for reliable transaction processing, image pre-fetching and migration, and so on. Many add additional features like proprietary report retrieval and reporting worklists.

## Electronic Medical Record

There are important lessons to be learned from PACS when adding standardized support for reporting. This is particularly true when one is working towards the holy grail of implementing a completely electronic medical record.

There are various terms and acronyms that are used in this regard that seem to be difficult to define precisely or distinguish. There is the Computer-based Patient Record (CPR), the Electronic Health-care Record Architecture (EHCRA), the Electronic

---

<sup>72</sup> Though such “external” workstations are often necessary to provide advanced features like 3D reconstruction, image fusion and analysis, etc.

Health Record (EHR), the Electronic Medical Record (EMR), the Electronic Patient Record (EPR), the Clinical Data Management (CDM) system, and so on.

Some of the key distinctions seem to be:<sup>73</sup>

- ▶ whether an existing paper process is simply replicated electronically or additional means of structuring and accessing information is provided
- ▶ whether or not the focus is on information about the patient per se (that is, beyond the confines of one health-care enterprise and incorporating multiple records in a longitudinal manner)
- ▶ whether or not the record is confined to conventional health-care encounters, or is more general and includes non-traditional as well as preventative care information

There is an enormous amount of effort being expended in approaching this holy grail, both in terms of commercial and academic systems that attempt to deliver it, and standards that purport to support it. As one might expect, many such systems make use of a web-centric approach for the delivery of content of the record to the user's desktop. Security features and concerns about privacy are a major concern for any record that is accessible outside the confines of an individual enterprise.

Where does DICOM, and in particular structured reporting, fit in this morass? As has been repeatedly emphasized, DICOM is a boundary standard not an architecture. It excels at providing support for the encoding of objects (such as images, waveforms and documents) that are transmitted between devices. Accordingly, in any more complete patient record system it makes sense to use DICOM as the means of obtaining images, waveforms and structured reports that pertain to them. It may also be appropriate to internally archive such DICOM objects, both in order to preserve their full fidelity, to search them for specific content, and to be able to verify digital signatures.<sup>74</sup>

Whether it makes sense to deliver DICOM objects to the user's desktop is another matter. In the case of images that the user needs to subsequently manipulate interactively in order to measure or adjust contrast, it probably does sometimes make sense to deliver DICOM data sets, say to a browser plug-in or applet. For simpler pre-formatted images a consumer format may be preferable.<sup>75</sup> In the case of reports it

---

73. See also:

- ▶ "<http://www.medrecinst.com/caregiver/applications.shtml>"
- ▶ "<http://stills.nap.edu/html/computer/>"
- ▶ Institute of Medicine. *The Computer-Based Patient Record: An Essential Technology for Health Care, Revised Edition (1997)*. National Academy Press. ISBN 0-309-05532-6. "<http://books.nap.edu/books/0309055326/html/1.html>"
- ▶ CEN TC251 prENV 13606. *Electronic Health-care Record Architecture*
- ▶ ASTM E1384-99. *Standard Guide for Content and Structure of the Electronic Health Record*

74. Digital signatures are discussed in the chapter on "Security: Privacy and Authenticity."

75. A more sophisticated approach is to deliver sections of an image at an appropriate spatial and contrast resolution appropriate to the display's capabilities "on demand," such as is advocated by Paul Chang. This method is referred to as a "dynamic transfer syntax."

almost certainly does not make sense to deliver DICOM structured report objects to the desktop. Rather, they need to be rendered and transmitted in a more appropriate form, like HTML or PDF, or perhaps XML with an XSL-T style sheet if web browsers ever really do incorporate support for the W3C's XSL recommendation.<sup>76</sup>

In this manner, the boundaries of a complete record system can be defined, both in terms of inputs (sources of objects) and outputs (objects and formats sent to the user's desktop). Even integration of diverse sources of input remains hard, particularly when the terminology used to describe concepts is not harmonized, or when information models differ. What remains to be defined is the system that manages the information and enacts the so-called "business logic." Whether such a system is amenable to standardization, and can be constructed from off-the-shelf components, or whether it needs to be monolithic and proprietary, remains to be seen.

In conclusion then, DICOM is only one, relatively small, piece of the very complex puzzle that is the completely electronic medical record. Though new kinds of objects together with management and security services may increase the penetration of DICOM, and more creative uses for DICOM transactions may be found,<sup>77</sup> other pieces of the puzzle need to be provided in order to build a complete system.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ DICOM composite objects are instances of SOP classes, and are assigned unique identifiers (UIDs)
- ▶ DICOM composite instances are normally persistent beyond the scope of their transmission; they are "documents" rather than "messages;" they can be used transiently under a message oriented paradigm if discarded
- ▶ the content and management information encoding in a structured report cannot be altered without creating a new instance; in particular, amendments or revisions are new documents
- ▶ specific rules govern the references between duplicates and versions of a structured report; all identical documents and prior versions are referenced
- ▶ structured reports are encoded with completion and verification status
- ▶ structured reports contain lists of other DICOM objects that are current or relevant information for the content of the report

---

76. As of the time of writing, existing browsers either don't support XML and XSL-T at all, or only an older draft proposal that is completely different from the final recommendation. This comment is made not to denigrate the early adopters, but to warn of potential incompatibilities.

77. This is particularly true of the use of DICOM beyond one physical site (which has been the predominant use in the past). With appropriate security in place, DICOM services can also be used for distributed updates between sites and enterprises, integration of multiple sources of trans-coded information such as lab, pathology, history and physical, and as a mediator of snapshots of information to and from local and referring doctors.

- ▶ the existing DICOM composite object query-retrieve mechanism can be used to find structured reports as well; there are some limitations that are a consequence of structured reports making extensive use of nested sequences; broader support for the existing relational rather than the baseline hierarchical query model would make structured reports and referenced objects easier to locate
- ▶ management of the workflow for creating and distributing reports would be desirable; the standard is being extended to provide interpretation worklists along similar lines to the existing modality worklist and performed procedure step SOP classes
- ▶ designing an architecture to support reporting applications is beyond the scope of a standard for communication across the boundaries between devices
- ▶ DICOM will likely play a small but key role in the completely electronic medical record of the future, serving to standardize sources of images, waveforms and related reports

# Templates: Bringing order to chaos ...

One of the great things about DICOM Structured Reporting is its tremendous flexibility. It has the power to describe a tree of information in almost any way the creator can imagine. This flexibility is also one of the scariest things about SR. Without guidance, different report creators can and will produce documents with different form and structure to express the same content. Consumers of reports, such as rendering and database applications, may be more effective, or indeed feasible at all, if there are some constraints on this flexibility.



**Not yet  
standardized.  
Final standard  
may differ  
substantially.**

## What is a Template?

Enter “templates.”<sup>1</sup> A template is simply a pattern of SR content that suggests or constrains concept names, relationship types, value types, and value sets for a particular application. A template can be as simple as a small pattern to describe the characteristics of a lesions, or as complex as an entire SR document tree for a specific reporting application. In some cases, a template that applies to an entire SR document may form the basis for the definition of an application specific SOP class.<sup>2</sup> In other cases, it may be used with one of the general purpose SR Storage SOP classes (Basic Text, Enhanced or Comprehensive). In the latter cases, though such a template does not

1. *Or more precisely, templates will enter “real soon now.” It was originally expected that templates would be included in Supplement 23, but they were not ready in time. Indeed, at the time of writing this, Supplement 53 on Templates and Context Groups has not yet been sent for public comment, much less ballot. Fortunately, there now seems to be consensus on the basic intent and layout of templates. Further delay is largely a consequence of implementing the SNOMED agreement and resolving outstanding issues on observation context. The content of this chapter is either sufficiently general as to be independent of the supplement, or where specifics are necessary, is based on revision 21 of Supplement 53, dated 3rd October, 2000.*
2. *The proposed Mammography CAD SR Storage SOP class described in Supplement 50 is an example of this category.*

constrain conformance and is not negotiated on the network, its use may still facilitate improved performance of a report consuming application.<sup>3</sup>

Templates are analogous to the modules and IODs of conventional composite objects in that they describe what content is encoded, provide definitions for the “attributes” that comprise the content, and define constraints on the values that may be used. For example, a typical modality specific module from PS 3.3 looks something like the extract shown in Table 7.

TABLE 7. Mammography image module extract

Attribute Name	Tag	Type	Description
Positioner Type	(0018,1508)	1	Enumerated Values: MAMMOGRAPHIC, NONE
Positioner Primary Angle	(0018,1510)	3	Position in degrees of the X-Ray beam in the coronal anatomical plane as if the patient were standing where movement of the X-Ray source from right to vertical is positive, and vertical is zero.
Positioner Secondary Angle	(0018,1511)	3	Position in degrees of the X-Ray beam in the sagittal anatomical plane as if the patient were standing where movement of the X-Ray source from anterior to posterior is positive, and vertical is zero.
Image Laterality	(0020,0062)	1	Laterality of the region examined. Enumerated Values: R = right L = left B = both (e.g. cleavage)
Organ Exposed	(0040,0318)	1	Organ to which Organ Dose (0040,0316) applies. Enumerated Value: BREAST Note: In the Mammography IOD, Organ Dose (0040,0316) refers to the mean glandular dose.

Notice in this example that each attribute:

- ▶ is uniquely specified by a tag consisting of a group and element
- ▶ is given a name
- ▶ is given a type which specifies whether the attribute must be present and have a value (1), must be present but may be zero length if unknown (2), or is optional (3)
- ▶ has a precise description of its meaning which includes limits on what values may be used, and in some cases, the interaction with other attributes

3. *It is important to remember that templates themselves are not units of conformance nor are they negotiated on the network during association establishment. For a template to be thus applied requires an equivalent SOP class. Though as will be described later, the identifier of a template is encoded in the SR content tree when it is used, a rendering application is still required to render all content unambiguously, regardless of whether or not a particular template has been used.*

- ▶ does not have a value type specified - allowable value representations are defined in PS 3.6

Templates need to contain similar information in order to define the structure and form of content items in the SR tree. Like module tables, templates are defined in tables with rows corresponding to individual “attributes.” An example of a hypothetical template is shown in Table 8.

**TABLE 8.** Hypothetical template for mass characteristics

NL	Rel with Parent	VT	Concept Name	VM	Req Type	Condition	Value Set Constraint
1	-	CODE	EV(,,"Shape")	1	M	-	DCID nnn
2	-	CODE	EV(,,"Margin")	1	M	-	DCID nnn
3	-	CODE	EV(,,"Density")	1	UC	IFF row 4 absent	DCID nnn
4	-	NUM	EV(,,"Density")	1	UC	IFF row 3 absent	units=DCID nnn
5	-	NUM	EV(,,"Diameter")	1	U	-	-
6	-	IMAGE	DCID nnn (purpose of reference)	1-n	U	-	-

Each row of a template defines:

- ▶ the nesting level of the content item, indicated by “>” signs<sup>4</sup>
- ▶ the relationship that the content item has with its parent
- ▶ the value type of the content item
- ▶ the concept name of the content item, either as a specific coded entry tuple, or the identifier of a context group that lists alternative names
- ▶ the number of times the content item may be repeated (the value multiplicity)
- ▶ conditions on the presence or absence or value set of the content item
- ▶ constraints on the value of the content item, such as the value set (context group) for a code value or the units of a numeric measurement

The nesting level and relationship columns deserve further explanation. A single table may describe one level of “nesting” in the SR tree, in which case it is essentially a flat list of content items, and the nesting level will always be blank. The relationship of each content item with the single parent of the entire template is defined in one of two ways:

- ▶ the “relationship with parent” field is left blank, in which case it must be specified wherever the template is invoked (included in another template)
- ▶ the “relationship with parent” field is specified, in which case it is always used, no matter where the template is invoked

4. This convention has been adopted to be consistent with sequence item nesting in module tables.

In both cases, the specification for all content items in a single level template must be the same: either all rows must have a relationship specified or none may.

Some specific features illustrated by the template in Table 8 should also be noted:

- ▶ various abbreviations are used to introduce concept names, condition arguments and value sets
- ▶ for concept names and coded values with a single term, Enumerated Value (EV) and Defined Terms (DT) are specified as a tuple (code value, coding scheme designator, code meaning)
- ▶ when a choice of terms may be used, context groups are specified by their numeric identifiers as baseline (BCID), defined (DCID) or enumerated (ECID)<sup>5</sup>
- ▶ when different value types may be used as alternatives for the same concept name, two or more rows are defined; these are specified as mutually exclusive by using conditions<sup>6</sup>

The “relationship with parent” field has been left blank for all rows which means, as has been described above, that the relationship to be used is defined where the template is invoked.

Table 9 is an example of another single level template that invokes the mass description template *at the same nesting level*.

TABLE 9. Hypothetical template that invokes mass characteristics as siblings

NL	Rel with Parent	VT	Concept Name	VM	Req Type	Condition	Value Set Constraint
1	-	TEXT	EV(,,"Description")	1	M	-	-
2	-	CODE	EV(,,"Finding")	1	M	-	EV(,,"Mass")
3	HAS PROP	INCLUDE	ETID(Table 8)	1	M	-	-

Notice that the relationship to be used has been specified where the template is invoked. Since the nesting level field is blank, the template specified in the “include” directive is invoked not as a child but at the same nesting level. The content items specified in the included template become siblings of the content items of the invoking template. The “include” directive is specified in the value type field, which is overloaded for this purpose. The template to be invoked is specified either as a defined or

5. For further explanation of the distinction between baseline, defined and enumerated context groups see the chapter on “Codes and Controlled Terminology.”
6. The mechanism of specifying arguments to the conditions remains to be fully defined. In simple examples, references to row numbers or other concept names suffices. In more complex templates with more complex conditions, arguments capable of specifying ancestors and descendants across more than one nesting level may be needed. It may also be necessary to distinguish between arguments that are defined by the template as opposed to what is actually instantiated. A mechanism along the lines of the XPath specification, extended to support tuples defining coded entries, might be useful.

an enumerated template ID (DTID or ETID) in the concept name field which is also overloaded.

Alternatively, a template may define multiple nesting levels. In the example in Table 10, the same template is used to describe characteristics of a mass, but it is invoked such that its content items become children of a node in the invoking template.

**TABLE 10.** Hypothetical template that invokes mass characteristics as children

NL	Rel with Parent	VT	Concept Name	VM	Req Type	Condition	Value Set Constraint
1	-	TEXT	EV(,,"Description")	1	M	-	-
2	-	CODE	EV(,,"Finding")	1	M	-	EV(,,"Mass")
4	> HAS PROP	INCLUDE	ETID(Table 8)	1	M	-	-

A final permutation of this example is to fully expand the included macro in place and define one template that contains multiple nesting levels, as shown in Table 11. Note that the templates in Table 10 and Table 11 result in exactly the same tree when expanded.

**TABLE 11.** Hypothetical template with multiple nesting levels

NL	Rel with Parent	VT	Concept Name	VM	Req Type	Condition	Value Set Constraint
1	-	TEXT	EV(,,"Description")	1	M	-	-
2	-	CODE	EV(,,"Finding")	1	M	-	EV(,,"Mass")
3	> HAS PROP	CODE	EV(,,"Shape")	1	M	-	DCID nnn
4	> HAS PROP	CODE	EV(,,"Margin")	1	M	-	DCID nnn
5	> HAS PROP	CODE	EV(,,"Density")	1	UC	IFF row 6 absent	DCID nnn
6	> HAS PROP	NUM	EV(,,"Density")	1	UC	IFF row 5 absent	units=DCID nnn
7	> HAS PROP	NUM	EV(,,"Diameter")	1	U	-	-
8	> HAS PROP	IMAGE	DCID nnn (purpose of reference)	1-n	U	-	-

## Other Template Features

The examples so far have barely scratched the surface of what is possible with templates. Since the proposed extension to the standard to support templates is in a state of flux, there is little point in going into further detail here. Sufficeth to say, some of the other features of templates include:

- ▶ the ability to specify multiple levels of nesting
- ▶ the ability to specify recursive templates either by directly including oneself, or including another template that at some point includes an “ancestor” template

- ▶ the ability to specify “tail-end iteration” of templates by including oneself as the last row without further nesting

Clearly, templates have the potential to grow large and complex. It is anticipated that the “source” of the standard for templates will actually be defined in some formal syntax from which the printed standard will be generated. Making this source accessible would allow implementers to develop tools to automatically derive applications for generating, parsing and validating template-based reports.

## Templates and the Meaning of Codes

An important observation about the structure of templates is the absence of any definition or description of the terms and value sets used. Contrast this with the existing module tables in DICOM which have a detailed and explicit description field. Though this is not too much of a problem when the meaning of a concept or value is “obvious,” as in the examples in this section, there remains the potential for confusion or ambiguity. In order to keep the tables compact, the convention has been adopted that explanatory material, when necessary, will be appended as footnotes to template tables.

This approach will not work well for the description of the meaning of individual codes, since these may well be used repeatedly in many templates. Indeed the question of where the meanings of coded entries are defined is a major issue for the standard in general. It is clearly possible for the standard to specify with detail and precision any codes that are defined by DICOM. The precise meaning of strings for defined terms and enumerated values have only rarely been spelled out in the past. Now that codes for the same purpose are defined in the standard, they should probably have an accompanying description or explanation. In some cases the “code meaning” field (or “display name” in HL7 terminology) will be sufficient to explain the exact meaning of the code, but this will not always be the case.

An open question is what to do about codes drawn from external coding schemes such as SNOMED and LOINC. Not every such term has a precise definition, but it is probably inappropriate for DICOM to specify the meanings of other peoples’ codes.<sup>7</sup> This is one illustration of the price paid for harmonization and reuse of code sets, the potential for ambiguity and loss of precision.

## Template Authoring

Who writes templates? Essentially there are two types of input required in the design of a template, input from:

- ▶ those who understand the application domain, such as specialist physicians, technologists and scientists
- ▶ those who understand the standard and the implementation issues

---

7. *Not to mention that it is way too much work.*

Both groups need to work together to combine their experience into useful templates. In many cases, the result is either sufficiently general or sufficiently critical to interoperability that it needs to be incorporated into the standard itself. Such templates will need to go through the normal balloting and quality control procedures with the same rigor as would normally be applied to the design of a SOP class or IOD.

Given that there is the potential for an explosion of highly specific templates for particular clinical or modality domains, it has been suggested that other, non-DICOM, mechanisms arise for the development and distribution of templates.

What has been discussed so far is highly specific to DICOM structured reporting. Other groups like HL7, CEN TC 251 and ISO TC 215 are also working on similar ideas, though with applications well beyond the scope of diagnostic imaging. It remains to be seen to what extent such efforts can be effectively harmonized, and whether or not it is worthwhile to try.

## Templates and Presentation

If structured reports were to be generated according to widely accepted templates, then parsing and presentation would be considerably simplified.

A particularly important application of templates is to define a pattern of semantically meaningful content that can be matched with a “style sheet” to drive presentation. For example, if an application were to “know” that a coded “finding” with a value of “mass” is always decorated with children one level deep which specify the characteristics of the mass either as coded or numeric value types, then a rendering engine could use a pattern which matched that sub-tree to substitute a meaningful sentence. Given a report fragment of the form:

```
<CODE:(,, "Finding")=(,, "Mass")
  <has properties CODE:(,, "Shape")=(,, "Round")>
  <has properties CODE:(,, "Margin")=(,, "Irregular")>
  <has properties CODE:(,, "Density")=(,, "Fat")>
  <has properties NUM:(,, "Diameter")="1.5"(,, "cm")>
```

which is structured according to such a template, the rendering engine might generate:

```
A 1.5 cm diameter mass is found, which has an irregular margin, a round shape and a fat density.
```

If everyone was to use the same template to describe the characteristics of a mass, style-sheets that performed such a transformation would be quite feasible, not to mention potentially interchangeable. In the absence of such templates, one might only be able to produce a literal rendition of the tree, in order to be safe and avoid introducing ambiguities.

A promising approach to transformation of SR trees using XSL-T is discussed further in a later chapter on “Trees, Traversal and Transformation.” The combination of SR templates and XSL-T style sheets has enormous potential to improve the readability of natural language documents generated from structured content.

## Template Proposals for Common Patterns

Even for a single domain, that of reporting diagnostic images, one can imagine a vast range of potential patterns, both for the overall outline and structure of the document as well as for often repeated fragments. Many people and organizations are working on “ontologies”<sup>8</sup> for such documents, and what follows consists only of a few examples.

Some of the templates that are needed for diagnostic image reporting include:

- ▶ overall outlines of documents
- ▶ classification and description of procedures, findings, conclusions and recommendations
- ▶ descriptive patterns for common findings, both general, such as mass, and specific such as destructive bone lesions and space-occupying intracranial lesions
- ▶ measurement patterns for linear distance, area and volume estimation, with appropriate constraints on measurement units as well as spatial coordinates and image references
- ▶ patterns for negation (i.e. “no mass” rather than “mass”)

Rather than providing formal template definitions, which are dry and boring to read as well as being premature,<sup>9</sup> the types of patterns that may be appropriate are illustrated by examples.

## Diagnostic Imaging Report Outline

It is probably extremely foolish to begin by attempting to specify something that is subject to each radiologist’s individual style and taste. Nevertheless, no matter what names are used for individual sections, or in what order they are specified, radiology reports usually contain something similar to what is illustrated in this example.

```
<CONTAINER: (,, "Report title")>
  <contains CONTAINER: (,, "Indication")>
  <contains CONTAINER: (,, "Request")>
  <contains CONTAINER: (,, "Procedure")>
  <contains CONTAINER: (,, "Previous findings")>
  <contains CONTAINER: (,, "Findings")>
  <contains CONTAINER: (,, "Differential diagnosis")>
  <contains CONTAINER: (,, "Conclusions")>
  <contains CONTAINER: (,, "Recommendations")>
  <contains CONTAINER: (,, "Diagnosis codes")>
```

---

8. The word “ontology” is used here because it is in vogue amongst those defining outlines for documents. *Ontology* is a pretty general word that Webster’s defines as “a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system.” See “<http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>” for an interesting discussion of the subject.

9. Premature that is, in the sense that the definition of templates in DICOM has not yet been completed.

```
<contains CODE:> *
```

In the example, content items that occur once are shown as in previous chapters, Content items that may occur zero or more times have an "\*" appended. In other examples those content items that may occur one or more times have a "+" appended. When an item may occur zero or one times a "?" is appended.

Whether the more deeply nested content is conveyed as plain text, or as coded or numeric entries, does not affect the overall outline of such a report. Embedded image references are likely to be common and are worthy of further attention.

```
<CONTAINER:(,,"Findings")>
  <contains TEXT:(,,"Finding")="... stuff ..."> *
  <contains IMAGE:(,,"Purpose")=(class,instance)>
  <contains TEXT:(,,"Finding")="... more stuff ..."> *
```

In such an instance one would likely want to use a "continuous" value for the continuity of content flag of the enclosing findings container.

In this example, the SR document was organized with the same outline as a traditional plain text or printed report. A completely different approach might be to ignore any presentation issues and organize the same semantic content according to how conclusions were inferred from compound findings, which in turn were inferred from atomic findings, and so on.

```
<CONTAINER:(,,"Recommendations")>
  <inferred from CONTAINER:(,,"Conclusions")> *
  <inferred from CONTAINER:(,,"Findings")> *
```

Such an organization might convey similar meaning but without implicit assumptions about what "conclusion" and "recommendation" actually mean. It would probably be much more difficult to render into something presentable to a reader.

## Classification of Procedures

If it is difficult to reach consensus on the choice of section headings for a document, it is well nigh impossible to standardize procedure codes. However, as long as the choice of codes to use is left open,<sup>10</sup> the basic elements that are useful to describe a procedure can be defined, as in the following outline:

```
<CONTAINER:(,,"Procedures")>
  <contains CONTAINER:(,,"Procedure")> +
    <has properties CODE:(,,"Procedure type")=(,,"")>
    <has properties DATETIME:(,,"Procedure date")="">
    <has properties PNAME:(,,"Performing physician")="">
    <has properties PNAME:(,,"Technologist")=""> *
    <has properties PNAME:(,,"Nurse")=""> *
```

<sup>10</sup> And as was made apparent in the chapter on "Codes and Controlled Terminology," there is certainly a plethora of such coding schemes, not to mention the many site-specific coding schemes.

In many cases, rather than designating a separate section of a document to describe the procedure, it may be more appropriate to use observation context, as discussed in the chapter on “Context: Who did what to whom, when...”. However, since observation context is “inherited,” if it is to apply to the entire tree it must be directly attached to the root node. In other words, one cannot just substitute “has obs context” relationships for the “has properties” relationships in the example shown above. This is one situation in which the desire to create a document that faithfully encodes the outline of a traditional document conflicts with the mechanisms available for conveying semantics rather than presentation.

## Descriptions of Common Findings

As was discussed in an earlier section of this chapter, using standard patterns for the descriptions of common findings greatly facilitates presentation. One example already used is the description of a mass:<sup>11</sup>

```
<CODE:(,, "Finding")=(,, "Mass")
  <has properties CODE:(,, "Shape")=> ?
  <has properties CODE:(,, "Margin")=> ?
  <has properties CODE:(,, "Density")=> ?
  <has properties NUM:(,, "Diameter")=(,, "cm")> ?
```

A template for the description of a mass could specify only the concept names and leave the value sets for each concept undefined, or it could specify baseline, defined or enumerated context groups. It is most useful to specify defined context groups, so that all users would use the same codes for the same values, but would be free to extend the list as necessary. For example, one might specify as codes in a defined context group the following values for the shape of a mass:

```
Round
Oval
Lobular
Irregular
```

A user of the template could never substitute their own equivalent code for “round” but they could extend the list with a code for “hexagonal” if that were appropriate.

This template includes a numeric measurement for diameter. Templates should constrain the units that may be used for numeric measurements. It doesn’t make sense to use units of area for measurements of linear distance. It also doesn’t make sense to use units that are too small or too large for the application, such as microns or miles for the diameter of a lesion on an X-ray. It is best if agreement can be reached on a single choice of unit, but more commonly a range of units including millimeters, centimeters and inches is necessary to avoid upsetting peoples’ sensibilities. This topic will be revisited when templates for measurements are discussed.

11. The concept names and value sets described here for the general description of a mass are the same as those specified in the ACR Breast Imaging Reporting and Data System (BI-RADS), 3rd edition. A good many of the BI-RADS terms are useful beyond the scope of mammography.

In many cases, the descriptive features applicable to a “general purpose” finding are also be applicable to a more “specialized” finding, but with constrained value sets or additional attributes. Unfortunately as currently proposed, the template documentation mechanism doesn’t provide an ability for “inheritance” of descriptions to build more specialized templates from general templates. The inclusion mechanism can be used to achieve some reuse, but it is not “parameterized.”

Some templates for the description of features may be highly domain specific. For example, a bone lesion on a radiograph might be described with a template such as:<sup>12</sup>

```
<CODE:(,, "Finding")=(,, "Bone lesion")
  <has properties CODE:(,, "Location")=
    "Epiphyseal"| "Metaphyseal"| "Diaphyseal"> +
  <has properties CODE:(,, "Multiplicity")=
    "Single"| "Multiple">
  <has properties CODE:(,, "Distribution")=
    "Focal"| "Multifocal"| "Diffuse">
  <has properties CODE:(,, "Density")=
    "Lucent"| "Sclerotic"| "Mixed">
  <has properties CODE:(,, "Periosteal reaction")=
    "Absent"| "Solid"| "Lamellated"| "Sunburst"
    | "Codmans triangle"| "Complex">
  <has properties CODE:(,, "Soft tissue mass")=
    "Present"| "Absent">
  <has properties CODE:(,, "Soft tissue calcification")=
    "Absent"| "Dystrophic"| "Amorphous"| "Other">
  <has properties NUM:(,, "Diameter)=(,, "cm")>
```

The use of such a template does not preclude the implementer or user from adding other features if a lesion warrants it. It is better from a rendering perspective if such additional features are grouped and highlighted as such, however, so that a style sheet can handle them with a default rule. For example, one could extend the previous template with a finding that is highly specific to a particular disease, hyperparathyroidism, as follows:

```
<CODE:(,, "Finding")=(,, "Bone lesion")
...
  <has properties CONTAINER:(,, "Other features")
    <contains CODE:(,, "Subperiosteal resorption")=
      "Present">
...

```

In some cases it is difficult to avoid a pattern where the entry is always a coded concept with a “present/absent” or “yes/no” value set and only the “present” or “yes” choice is ever used. This is a consequence of the name-value pair structure of SR content items. Other than containers (which are clearly inappropriate for anything other than specifying containment or headings), there is no value type to convey a “name” alone without a corresponding “value.”

12. Many of the terms used here are derived from Richardson M. “Approaches To Differential Diagnosis In Musculoskeletal Imaging.” 1994. “<http://www.rad.washington.edu/mskbook/>”.

It is very often necessary for mandatory attributes to provide some sort of “escape” code to indicate a value is unknown, other or complex. For example in the preceding template, soft tissue calcification may be difficult to characterize, hence there is an “other” value permitted.

Even though this template example is highly specific to the clinical domain of musculo-skeletal radiology, it is not necessarily modality specific. The same template might be reused for CT findings, though some additional features might be required. While there is no inheritance mechanism for template documentation, the inclusion feature can be used to add findings, particularly if they are contained in a separate single level template that is a list of atomic features.

For example, for CT findings one might write:<sup>13</sup>

```
<CODE:(,,“Finding”)=(,,“Bone Lesion”)  
  %include “bone lesion features template”  
  <has properties CODE:(,,“Fluid Level”)=  
    “Present”|“Absent”>  
  <has properties NUM:(,,“Density)=(,,“HU”)>  
    <inferred from SCOOD: (...)>  
    <selected from IMAGE: (CT image,...)>
```

In the case of CT, it is not unusual to quote measurements of the density of a region of interest (ROI). In the basic template there was also a measurement of the diameter of the lesion, though without an accompanying set of spatial coordinates or an image reference. Since there may be many feature description templates that involve some type of measurement, it makes sense to factor out templates for measurements that can be reused.

## Measurement Templates

Measurements are either general or specific to a particular imaging technique. For example, there are diameters of objects in general, and there are fetal ultrasound biparietal diameters used for estimation of gestational age.

Most measurements in diagnostic imaging are structural, that is they are some estimate of an object’s size. Other measurements are of image intensity, which usually reflects some physical property that is the basis of the imaging technique. For example, CT image intensity is proportional to the density of the object and is measured in Hounsfield Units. The image intensity of a normal MR image is a function of relaxation times and proton density. MR images may also be phase or velocity encoded. For some modalities and techniques, functional information can be obtained from image intensity. For example one may measure image intensity over time after injection of a contrast agent, or the counts received by a gamma camera after injection of a radio-nuclide.

Both general and specific measurements of structural, physical and functional properties are amenable to description by templates.

---

13. Inventing a new shorthand convention for the inclusion of another template as one goes.

There are a handful of different categories of measurements, including measurements of:

- ▶ the linear distance between two points
- ▶ the area of an object computed from its perimeter or estimated from two perpendicular linear distances
- ▶ the volume of an object computed by integrating areas determined from contiguous parallel slices

These structural measurements are differentiated by:

- ▶ what structure is being measured (e.g. a mass, a fetal head, a kidney)
- ▶ what feature is being measured (e.g. an area, a diameter, a volume)
- ▶ what units are used to describe the value

One can envisage a general “structural measurement” template that specifies attributes for these parameters, as well as a mechanism for referencing the spatial coordinates and images from which the measurement was derived. For example:

```
<NUM:(,,,"Diameter"| "Area"| "Volume")= ... (,,,"units")>
  <has concept mod CODE:(,,,"Structure")=...>
  <inferred from SCOOD: (...,...)>
    <selected from IMAGE: (CT image,...)>
```

In this extremely simple pattern, the feature is encoded as the concept name of the numeric value type, which is then modified by a concept modifier which defines the anatomical structure being measured. This approach of constructing a post-coordinated descriptor is preferred, because it is a very simple matter to add further modifiers to indicate the method, and so on. For example:

```
<NUM:(,,,"Diameter")= 79.2 (,,,"mm")>
  <has concept mod CODE:(,,,"Structure")=(,,,"Fetal head")>
  <has concept mod CODE:(,,,"Method")=(,,,"Biparietal")>
  <has concept mod CODE:(,,,"Technique")=(,,,"HadLock")>
```

Some measurements are aggregates, derived from more elementary measurements. An indication of how the aggregate measurement was derived is often required, together with the values and source of the elementary measurements:

```
<NUM:(,,,"Diameter")= 79.2 (,,,"mm")>
  <has properties CODE:(,,,"Derivation")=(,,,"Mean")>
  <has properties NUM:(,,,"Number of samples")= 2 >
  <inferred from NUM:(,,,"Diameter")= 79.4 (,,,"mm")>
    <inferred from SCOOD: (...,...)>
      <selected from IMAGE: (CT image,...)>
  <inferred from NUM:(,,,"Diameter")= 79.0 (,,,"mm")>
    <inferred from SCOOD: (...,...)>
      <selected from IMAGE: (CT image,...)>
```

Specific templates need to be defined for the type feature being measured, in order to constrain both the form of the spatial coordinates as well as the units that are permitted. The following three examples specialize the general template shown earlier, into

templates for diameters, areas and volumes. This is a template for a diameter specified by its end-points and in units of millimeters:<sup>14</sup>

```
<NUM:(,, "Diameter")= ... ("mm", "UCUM", "millimeter")>
  <has concept mod CODE:(,, "Structure")=..>
  <inferred from SCOORD: (MULTIPOINT, Sx, Sy, Ex, Ey)>
  <selected from IMAGE: (image, ...)>
```

This is a template for an area specified by a circle, an ellipse or a polyline (which is implicitly closed), in units of square millimeters:

```
<NUM:(,, "Area")= ... ("mm2", "UCUM", "square millimeter")>
  <has concept mod CODE:(,, "Structure")=..>
  <inferred from SCOORD: (CIRCLE|ELLIPSE|POLYLINE, ...)>
  <selected from IMAGE: (image, ...)>
```

This is a template for a volume specified by a circle, an ellipse or a polyline (which is implicitly closed) that is applied to selected frames of a multi-frame image, in units of cubic millimeters:

```
<NUM:(,, "Volume")= ... ("mm3", "UCUM", "cubic millimeter")>
  <has concept mod CODE:(,, "Structure")=..>
  <inferred from SCOORD: (CIRCLE|ELLIPSE|POLYLINE, ...)>
  <selected from IMAGE: (multiframe image, ..., frames)>
```

This is a similar template for a volume derived from multiple sets of coordinates, each of which applies to a single frame image:

```
<NUM:(,, "Volume")= ... ("mm3", "UCUM", "cubic millimeter")>
  <has concept mod CODE:(,, "Structure")=..>
  <inferred from CONTAINER:(,, "Coordinates")>
  <contains SCOORD: (CIRCLE|..., ...) +
  <selected from IMAGE: (single frame image, ...)>
```

In all likelihood, even though one unit of measurement with the same dimensions may easily be derived from another, users will expect to be able to specify the units they prefer and to see them rendered in the units they choose. Accordingly, templates for measurements may specify the choice of units through defined or enumerated context groups, such as those in Table 12.

Another factor to consider when defining templates for measurements is how they may be applied in combination with temporal coordinates. For example, in the very specific instance of estimating left ventricular volume from an outline drawn around

14. One of the many possibilities for using spatial coordinates to designate the end-points of a line has been chosen here. See the chapter on "Content Items: Concept Names and Values" for other alternatives.

The measurement templates proposed in Supplement 50 Mammography CAD suggest a slightly different approach. The name of the measurement is indicated by a text value type that is a sibling of the numeric measurement, rather than a coded concept modifier. Rather than specify diameters, a general "linear distance" template is proposed in the form of a "path" that may be end-points for a simple line, a circle or an ellipse for a circumference, or a polyline for a more general form of perimeter.

**TABLE 12.** Measurement units for distance, area and volume

Code Value	Coding Scheme Designator	Coding Scheme Version	Code Meaning
<i>Linear Distance:</i>			
um	UCUM	1.4	micrometer
mm	UCUM	1.4	millimeter
cm	UCUM	1.4	centimeter
[in_i]	UCUM	1.4	inch
<i>Area:</i>			
um2	UCUM	1.4	square micrometer
mm2	UCUM	1.4	square millimeter
cm2	UCUM	1.4	square centimeter
[sin_i]	UCUM	1.4	square inch
<i>Volume:</i>			
um3	UCUM	1.4	cubic micrometer
mm3	UCUM	1.4	cubic millimeter
cm3	UCUM	1.4	cubic centimeter
[cin_i]	UCUM	1.4	cubic inch

the contrast-filled inner margin of the ventricle on a single-plane projection X-ray angiography image, one needs a template similar to the following:

```
<NUM:(,, "Volume")= ... ("cm3", "UCUM", "cubic centimeter")>
  <has concept mod CODE:(,, "Structure")=(,, "LV")>
  <has concept mod CODE:(,, "Time")=(,, "end systole")>
  <has concept mod CODE:(,, "Method")=(,, "...")>
  <has concept mod CODE:(,, "Projection")=(,, "RAO")>
  <inferred from TCCORD:(...,...)>
    <inferred from SCOOD: (CIRCLE|...,...)>
      <selected from IMAGE: (XA image, ...)>
```

Alternatively, the frame number(s) be specified directly in a single image reference without recourse to temporal coordinates.

Finally, just a reminder that not all measurements are of size like distance, area and volume. For example, this is a template for the density of a region of interest on a CT image:

```
<NUM:(,, "Density")= "34.3"
  ("[hnsf'U]", "UCUM", "Hounsfield unit")>
  <inferred from SCOOD: (CIRCLE|ELLIPSE|POLYLINE, ...)>
  <selected from IMAGE: (CT image, ...)>
```

## Normal Ranges for Measurements

A common pattern used in measurements for medical applications is to specify not only the value, units and derivation of a measurement, but also what the “normal” or “reference” range for that measurement is,<sup>15</sup> and whether or not the measured value is within that range (normal) or outside it (abnormal).

```
<NUM:(,,,"some measurement")= 1.6 ("g/dl","UCUM","")>
  <has properties CODE:(,,,"Normality")=(,,,"Abnormal")>
    <inferred from NUM:(,,,"Lower limit")= 0.1 ("g/dl")>
    <inferred from NUM:(,,,"Upper limit")= 1.2 ("g/dl")>
```

## Negation Templates

In the chapter on “SOP Classes and Conformance,” the issue of conveying meaning in a clear and unambiguous manner was raised with respect to the issue of negation. Specifically, it is critical to be sure that a statement like “no metastases” is not misinterpreted as “metastases” because the rendering application is not able to detect the negation pattern being used. There are a number of approaches to this problem.

One approach is to use pre-coordinated codes, such that for every potential feature such as a “metastasis” there is a corresponding code that means “no metastasis.” While this would be safe, it is obviously not a solution that scales very well.

If pre-coordinated codes are not the answer, then a post-coordinated approach might work. One could define a concept modifier that indicated the presence or absence of a feature:

```
<contains CODE:(,,,"Finding")=(,,,"Metastases")>
  <has concept mod CODE:"Presence"="Absent">
```

In the absence (sic) of such a modifier, then the finding could safely be assumed to be present.<sup>16</sup> As was pointed out earlier, for this approach to work safely it would be crucial that applications did not “ignore” concept modifiers.

Another approach would be to add to any list of features an additional coded value type as a sibling to indicate negation. For example:

```
<CODE:(,,,"Shape")=(,,,"...")>
<CODE:(,,,"Margin")=(,,,"...")>
<CODE:(,,,"Density")=(,,,"...")>
...
<CODE:(,,,"Existence negation flag")=(,,,"Not present")>
```

This would be defined to mean that all concepts conveyed by the descendants of the enclosing content item would be negated.

15. The normal range is very often specific to the method used, the equipment used, and the state of calibration. Accordingly, it may be difficult to determine whether a measured value is normal or abnormal retrospectively without this additional information.

16. This default is necessary, because it would be clumsy and unnecessary to always require the modifier with a value of “present.”

Yet another approach is to define a container whose contents are not present. For example:

```
<CONTAINER: (,, "Absent features")>
  <CODE: (,, "Finding")=(,, "Metastases")>
```

The use of a concept modifier is probably the safest and most scalable mechanism. This is clearly an aspect of structured reporting on which consensus is urgently required for coded concepts to be used unambiguously.<sup>17</sup>

There are other related patterns of description that are similar to negation, but specific to a particular context. Radiologists are fond of stating that certain structures are “normal” or “clear.” For example:

```
The heart size is normal and the lung fields are clear.
No bony metastases are seen.
```

In some cases it is appropriate to use specific codes and in others a general negation pattern. One might encode the same meaning using coded value types as:

```
<CONTAINER: (,, "Findings")>
  <contains CODE: (,, "Structure")=(,, "Heart")>
    <has properties CODE: (,, "Size")=(,, "Normal")>
  <contains CODE: (,, "Structure")=(,, "Lungs")>
    <has properties CODE: (,, "Parenchyma")=(,, "Clear")>
  <contains CODE: (,, "Structure")=(,, "Bones")>
    <has properties CODE: (,, "Finding")=(,, "Metastases")>
      <has concept mod CODE: "Presence"="Absent">
```

Of course a more concise way of reaching the same outcome is to simply say:

```
<CONTAINER: (,, "Findings")>
  <contains CODE: (,, "Finding")=(,, "Normal")>
```

## Identifying Which Templates Are In Use

This chapter has discussed the importance of templates and proposed various templates for a number of useful patterns. How is an application that receives reports that have been constructed using templates to know which templates have been used and where?

In the cases where an entire SOP class is defined by a “master” template, such as the proposed Mammography CAD SOP class, the answer is clear. However, even in those cases, it is not always easy to identify which subordinate templates have been used lower down in the tree. This is also true of any template used within a structured report encoded as a generic SOP class instance.

Each content item conditionally contains a Template Content Sequence (0040,A504) attribute that contains a single item that is reminiscent of a code sequence macro. It is

---

17. The most recent draft of Supplement 53 revision 20, dated 3rd October, 2000, proposes a single modifier (on individual properties) of “presence” with values of “present” or “absent.”

required to be present if a template was used to define the content of the item. The content of the sequence item is defined by the template identification macro which includes the attributes shown in Table 13.

**TABLE 13.** Summary of Template Identification Macro

Attribute	Tag	Type
Template Identifier	(0040,DB00)	1
Mapping Resource	(0008,0105)	1
Template Version	(0040,DB06)	1C
Template Local Version	(0040,DB07)	1C
Template Extension Flag	(0040,DB0B)	1C
Template Extension Organization UID	(0040,DB0C)	2C
Template Extension Creator UID	(0040,DB0D)	2C

The idea is that if a template has been used to create a sub-tree rooted at a particular node, then that node will be decorated with template identification information. This works fine if a template is defined such that it consists of a single content item at the top nesting level, and all other content items at subordinate nesting levels. It doesn't work if the template consists of a list of items at the same nesting level. One could argue that the later case could be solved by decorating every node generated by using a particular template with the template's identification. Quite apart from the duplication, this doesn't work either, since content may be generated as a result of instantiating multiple nested templates. In other words one might have to decorate each node with multiple template identifiers.

Unfortunately, it isn't really possible, without extremely convoluted encoding mechanisms, to directly "recover" the identifiers of what templates were used. An analogy that programmers will understand is the use of macro pre-processors for generation of code. The C language pre-processor (cpp) is very flexible in its ability to convert instructions from one form into another, but it is impossible to "invert" the transformation, that is to recover the original code with the macros from the generated code that is fed to the compiler.

Fortunately, this doesn't much matter in practice. One of the primary roles of templates is to achieve consistency in the way in which the SR tree is organized. Given such a "well-organized" tree, it is not difficult to define patterns that can be used to match the anticipated tree structure, and trigger appropriate behavior. For example, one may define templates and corresponding patterns for rendering style sheets, and use a pattern matching engine rather than depend on explicit template identification information conveyed in the encoded data set. One of the beneficial side effects of not depending on explicit template identification information is that such pattern matching style sheets are equally applicable to appropriate content that was not generated using the template, but happens to have a similar organization.

## Summary

In this chapter, some of the basic concepts behind the idea of templates have been introduced. Some suggestions for templates for common applications have been shown as examples of how useful templates might be. No further detail can be specified until further development of the standard has been completed.

Some of the key points mentioned in this chapter include:

- ▶ templates are a mechanism for improving consistency and interoperability while using what is otherwise an essentially unconstrained framework
- ▶ templates specify the structure of a part of an SR tree
- ▶ templates specify constraints on the contents of part of an SR tree, in terms of choice of concept names, value types, value sets, and measurement units
- ▶ templates are analogous to the module tables used in conventional composite objects that define which attributes are used and with what values
- ▶ templates are potentially important for enabling the use of presentation style sheets that are based on consistent patterns of content
- ▶ in practice, it may be difficult to recover template identifiers from the encoded data set



# *Internationalization and Localization*

---

Contrary to popular belief, DICOM is not a US standard, nor is it strictly for applications in English-speaking environments. The DICOM Committee consists of vendors, trade organizations, users and members of professional societies from all over the world, including Asia and Europe. Though the standard is written in US English, implementations of the standard can use almost any language.

In the world of imaging, the priority for internationalization and localization<sup>1</sup> of implementations is to be able to handle people's names correctly. In the course of adding support for this application, DICOM has acquired the generic capability to represent any string in most languages.

The creation of localized forms of structured reports exploits this mechanism to its limits. Reports include a wealth of complex textual information, or coded information that needs to be rendered as localized text.

In this section, the basic mechanisms of encoding text beyond the default character set (which is US-ASCII) will be described. Alternative ways to handle standard codes that need localized meanings will be discussed, as well as the issue of simultaneous multi-lingual representations of code meanings and text. For a more thorough treatment of encoding text, especially Asian languages, consult Ken Lunde's authoritative bible on CJKV, which, though it doesn't describe the specifics of DICOM encoding, does address everything else and more.<sup>2</sup>

---

1. The term "localization" is used in this book to mean adapting an application to a country-specific environment, particularly with respect to language. Other people use the same term to mean adapting to a specific hospital or site, such as configuring site specific codes for concepts. This is not the use of the term here. The term "internationalization" implies making "localization" possible. See Uren et al. "Software Internationalization and Localization." Van Nostrand Reinhold 1993. ISBN 0-442-01-01498-8.

### Basic Text Encoding: Single Byte Character Sets

In the beginning, there was only the default character set, US-ASCII, more properly referred to in the world of international standards as ISO 646, and registered as ISO-IR 6. This is a single-byte seven-bit character set, which means that there are 128 characters available, including alphanumerics, control characters and punctuation. One ASCII character always occupies one byte, so a 64 character limit on the length of a string is equivalent to a 64 byte limit.

A seven-bit character set essentially “wastes” 128 possible encodings for characters, so one approach to adding more characters is to use the space available with the high bit set. Almost all of the other “single byte” character sets take this approach, keeping the lower 128 characters (more or less) consistent with ASCII, and using different sets of characters in the high 128 positions.

The most popular of these is ISO 8859-1. This is a character set which encodes all of the additional characters with accents and the like that are needed for representing Western European languages. This character set is also referred to by its international registry (IR) number, ISO-IR 100. The ISO 8859 family of standards includes support for a range of different national character sets, not just Latin alphabets, but others such as Cyrillic, Arabic, and so on. Most of these standards were originally created as ECMA standards.<sup>3</sup>

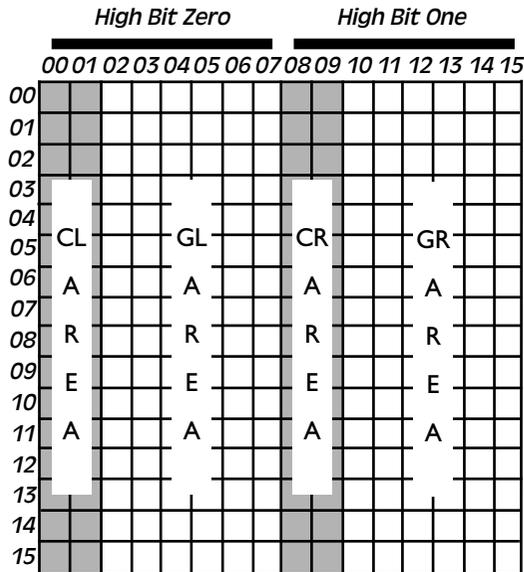


FIGURE 48. Areas of a Single Byte Code (as per ISO 2022)

2. CJKV stands for Chinese, Japanese, Korean and Vietnamese. The book is Lunde, Ken. “CJKV” O’Reilly 1999. ISBN 1-56592-224-7.

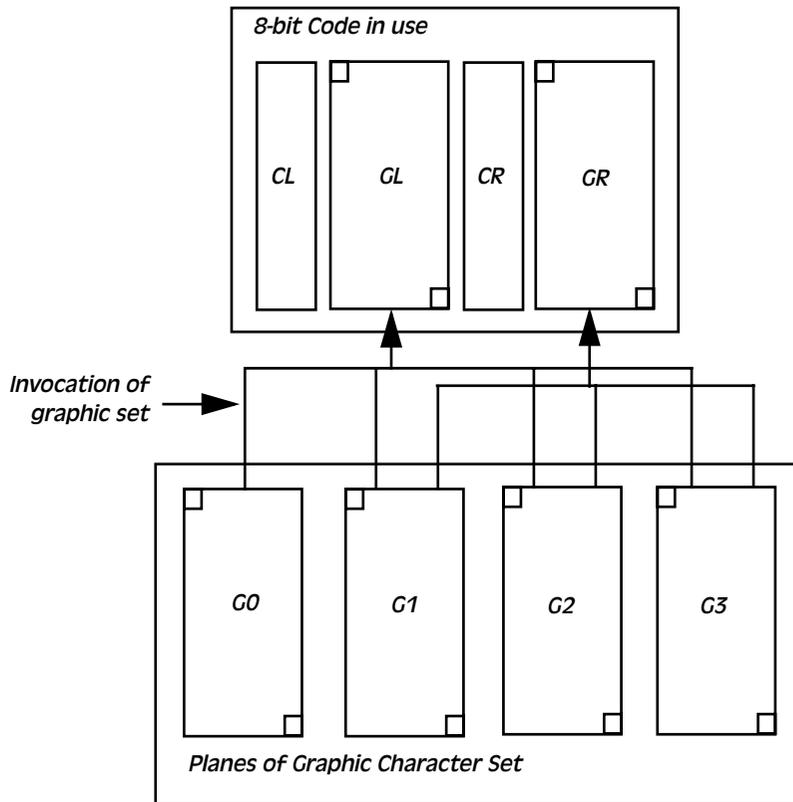


FIGURE 49. Structure of Single Byte Codes (as per ISO 2022)

It is beyond the scope of this book to go into a detailed explanation of how the ISO 8859 standards define various parts of the range of 256 values available in eight bits. However, it suffices to say that the “space” that is represented by an encoded single byte character is divided into “planes,” as illustrated in Figure 48. Character sets are described in terms of “code elements” that may be invoked in the “planes” of an encoded byte, as shown in Figure 49.

In DICOM, the “G0 element” of a character set is always invoked in the “left-hand part” or “GL plane” of an encoded byte, as opposed to the “G1 element” of a character set which is always invoked in the “right-hand part” or “GR plane” of an encoded byte. The GL plane consists of the graphic characters in the low half of the encoded byte, i.e. when the high bit is zero. The GR plane consists of the graphic characters in the high half of the encoded byte, i.e. when the high bit is one.<sup>4</sup>

3. ECMA stands for European Computer Manufacturers Association. The nice thing about ECMA standards is that they are free, whereas ISO standards cost an arm and a leg. Just go to “<http://www.ecma.ch/>” and either download what is needed, ask for a CD of many of their standards, or ask for them to send (free) a printed copy of the standards. Amazing.

In most cases, the usage in DICOM specifies that the G0 element invoked in the GL plane will be ASCII<sup>5</sup> aka ISO 646 aka ISO-IR 6, and further, that the GR plane will be either unspecified, or the G1 element specified by the ISO-IR number corresponding to one of the ISO 8859 standards. The choice is selected by using the attribute Specific Character Set (0008,0005), which is an optional attribute of the SOP Common Module of every DICOM object.

For example, to select the Western European Latin Alphabet No. 1, the defined term of “ISO\_IR 100”<sup>6</sup> is used in Specific Character Set to indicate that:

- ▶ the G0 element is ASCII aka ISO 646 aka ISO-IR 6
- ▶ the G1 element is ISO 8859-1 aka ISO-IR 100 aka (part of) ECMA 94

Similarly, to select the Latin/Greek Alphabet, the defined term of “ISO\_IR 126” is used in Specific Character Set to indicate that:

- ▶ the G0 element is ASCII aka ISO 646 aka ISO-IR 6
- ▶ the G1 element is ISO 8859-7 aka ISO-IR 126 aka ECMA 118

This “alphabet soup” of possible single byte character sets is probably best summarized in a table. Most of the character sets defined in this table share the characteristic that the G0 element is always ISO 646. The G1 element’s contents are filled by the character set specified.<sup>7</sup>

**TABLE 14.** Sources of Single Byte Character Sets (with G0 ISO 646)

Description	IR#	Defined Term	ISO Source	Other Source	G0 Source
Latin alphabet #1	100	ISO_IR 100	ISO 8859-1	ECMA 94	ISO 646
Latin alphabet #2	101	ISO_IR 101	ISO 8859-2	ECMA 94	ISO 646
Latin alphabet#3	109	ISO_IR 109	ISO 8859-3	ECMA 94	ISO 646
Latin alphabet #4	110	ISO_IR 110	ISO 8859-4	ECMA 94	ISO 646
Cyrillic	144	ISO_IR 144	ISO 8859-5	ECMA 113	ISO 646

4. In addition to the G0 and G1 planes used in DICOM, there are (theoretically) G2 and G3 planes in some character sets, as illustrated in Figure 49. These are never used in DICOM. Also defined are elements (C0 and C1) and planes (CL and CR) for control characters, which are not affected by the choice of character set in DICOM.
5. Strictly speaking, ISO-IR 6 refers to US-ASCII, the version of ISO 646 defined by ANSI X3.4-1968, as opposed to ISO-IR 4 which refers to British ASCII defined by BSI 4730.
6. Note that the defined term encoding in the DICOM attribute uses an underscore ‘\_’ rather than a hyphen ‘-’ to separate the “ISO” and the “IR”, since a hyphen is not allowed in the value representation of the attribute, which is Code String (CS).
7. Notice that in the standard the ISO-IR number is specified, but the relevant part of ISO 8859 is not; it just says “supplementary set of ISO 8859.” Here, the part of ISO 8859 is also shown (as well as the corresponding freely available ECMA standard). Not all registered character sets (with an ISO-IR number) correspond to ISO standards; some correspond to national standards. For example, Thai International Standard TIS 620 was recently added to DICOM, though there is not (yet) a corresponding part of ISO 8859. Obviously, other single byte character sets with ISO registry entries can be easily be added to DICOM on request.

**TABLE 14.** Sources of Single Byte Character Sets (with G0 ISO 646)

Description	IR#	Defined Term	ISO Source	Other Source	G0 Source
Arabic	127	ISO_IR 127	ISO 8859-6	ECMA 114	ISO 646
Greek	126	ISO_IR 126	ISO 8859-7	ECMA 118	ISO 646
Hebrew	138	ISO_IR 138	ISO 8859-8	ECMA 121	ISO 646
Latin alphabet #5	148	ISO_IR 148	ISO 8859-9	ECMA 128	ISO 646
Latin alphabet #6	???	-none-	ISO 8859-10	ECMA 144	ISO 646
Thai	166	ISO_IR 166		TIS 620	ISO 646
Katakana	13	ISO_IR 13		JIS X 201 <sup>a</sup>	JIS X 201

a. See “ftp://ftp.tiu.ac.jp/pub/jis/jisx0201/”.

The Latin Alphabets each contain characters to support multiple national languages, and there is considerable overlap between them. Most are derived from Latin Alphabet No. 1. For example, the only difference between Latin Alphabets 1 and 5 are that six Icelandic characters were replaced by the letters required for Turkish.

**TABLE 15.** Correspondence between Languages and Latin Alphabets

	Latin 1	Latin 2	Latin 3	Latin 4	Latin 5	Latin 6
Afrikaans			X			
Albanian		X				
Catalan			X			
Czech		X				
Danish	X			X	X	X
Dutch	X				X	
English	X	X	X	X	X	X
Esperanto			X			
Estonian				X		X
Faroese	X					X
Finnish	X			X	X	X
French	X		X		X	
Galician			X			
German	X	X	X	X	X	X
Greenlandic				X		X
Hungarian		X				
Icelandic	X					X
Irish	X				X	
Italian	X		X		X	
Latvian				X		X

TABLE 15. Correspondence between Languages and Latin Alphabets

	Latin 1	Latin 2	Latin 3	Latin 4	Latin 5	Latin 6
Lithuanian				X		X
Maltese			X			
Norwegian	X			X	X	X
Polish		X				
Portuguese	X				X	
Rumanian		X				
Sami (Lappish)				X		X
Serbo-croatian		X				
Slovak		X				
Slovene		X				
Spanish	X				X	
Swedish	X	X		X	X	X
Turkish			X		X	

In practice, many DICOM devices and applications support the ISO 8859-1 character set in addition to the default ASCII repertoire. As one can see from the table, this is sufficient for many of the European languages, without having to consider switching between different character sets.

There is currently defined in DICOM one single byte character set that does not use ISO 646 (ASCII) in the G0 element, and that is a single byte Japanese Katakana set, ISO-IR 13. In DICOM, when the defined term for Specific Character Set is “ISO\_IR 13”, then JIS X 201 is used, with Katakana in the G1 element, and Romaji<sup>8</sup> in the G0 element. As it happens, there isn’t a whole lot of difference<sup>9</sup> between ASCII and JIS X 0201 Romaji, so this usage is consistent with always having ISO 646 in the G0 element.

## Mixing and Matching: Code Extension Techniques

Being able to specify a single character repertoire for the entire data set is sufficient for many, but not all applications. The DICOM standard allows for the following possibilities through the use of the attribute Specific Character Set (0008,0005):

- ▶ the attribute is absent or zero length, in which case the “default character repertoire” is used, i.e. US-ASCII (ISO 646)

8. The Romaji plane is actually designated in the registry as ISO-IR 14, but in the DICOM usage, when ISO-IR 13 is invoked, ISO-IR 14 is as well.

9. The differences are that in JIS X 201, 0x5C is not a backslash ‘\’ but rather a yen symbol ‘¥’, and 0x7E is not a tilde ‘~’ but rather an overline. This is a very important difference from a DICOM perspective, since the 0x5C character is specified as the delimiter between multiple values in string value representations. This delimiter will appear as ‘¥’ when using ISO-IR 13.

- ▶ the attribute has a single value that is one of the recognized defined terms,<sup>10</sup> in which case all characters in all strings will use that character set (this is called the “replacement character repertoire”)
- ▶ the attribute has multiple values that are all recognized defined terms, in which case the first value specifies the initial character set at the beginning of each string (the “replacement character repertoire”), and “escape sequences” are used to switch different character sets (the “extended character repertoires”) into the plane specified by the escape sequence; the first value may be absent in which case it is equivalent to specifying the default character repertoire (ISO 646 or ISO-IR 6)

As a rather contrived example of the use of code extension techniques, consider how to send a string that contains english, french and katakana. Specifically the string contains the english for “english,” the french for “french” and the katakana for “japanese (language),” as follows:

```
“English\Français\ ニホンゴ ”
```

What character sets are available? The default character repertoire (ISO 646), any of the ISO 8859 family, and JIS X 201 Romaji can all be used to represent the English string. ISO 8859-1, -3 or -5 can be used to represent the ‘c’ with a cedilla in the French string. JIS X 201 Katakana is needed for the Japanese string.

The approach used in DICOM is to:

- ▶ signal in Specific Character Set which character sets may potentially occur
- ▶ use a *very limited subset*<sup>11</sup> of ISO 2022 “escape sequences” to switch between character sets in the middle of a string

To encode the string in the example, Specific Character Set might be set to:

```
“ISO 2022 IR 100\ISO 2022 IR 13”
```

Notice that one uses different defined terms when using code extension techniques, compared to using a single character set. Each defined term is preceded by “ISO 2022”. There is a corresponding ISO 2022 “shadow” for each one of the single byte character sets described earlier. Additionally there is an “ISO 2022 IR 6” defined term, so that one can re-invoke the default character repertoire (ISO 646) in the GL plane when necessary. For each defined term, two escape sequences are defined, one to invoke the G0 character set in the left-hand graphic plane (GL), and one to invoke

---

10. The standard is silent on what to do if a value of Specific Character Set is not a recognized defined term. This may happen when an older implementation encounters a recently added character set, or when a privately defined extension to the standard is in use. There is little a receiver can do in this case except hope that the character set is single byte and the G0 plane is ASCII. If the value of specific character set is of the form “ISO\_IR nnn” or “ISO 2022 IR nnn”, where nnn is not defined in the standard but is in the ISO registry, more sensible action can be taken by a very well informed application.

11. ISO 2022 is a very comprehensive standard that allows one to switch any element into any plane in a variety of ways. It defines all sorts of shift mechanisms, as well as support for single and multiple byte character sets. Only a very limited subset of these is needed in DICOM or, for that matter, is permitted. Only those escape sequences explicitly listed in DICOM may be used.

the G1 character set in the right-hand graphic plane (GR) (except IR 6, which only has a G0 element and therefore only one escape sequence).<sup>12</sup>

The important thing to remember is that G0 and G1 code elements are activated in the GL and GR planes independently. At any point in time, one may have the G0 element of any of the character sets (listed in Specific Character Set) active in GL, and the G1 element of the same or some other character set active in GR.

In practice, there is almost always a G0 element equivalent to ASCII in the GL plane. Of the character sets currently defined, all have a G0 element of ISO 646 (ASCII), except IR 14, which has Romaji (JIS X 201) instead.

The question of which character set is initially active has not yet been considered. The rule is that the character set in the first value of Specific Character Set (or if absent, the default character repertoire) shall be active at the beginning of each value of the string, and the beginning of each line.<sup>13</sup> Implicit in this statement is that if the character set specifies both G0 and G1 elements, then those will be invoked in the GL and GR planes respectively. If the character set (like the default repertoire) specifies only one or the other of the G0 or G1 planes, then only the corresponding GL or GR plane will be active, and the other will be *undefined*.<sup>14</sup>

In the example, since “ISO 2022 IR 100” is the first value of Specific Character Set, the additional characters of Latin Alphabet No. 1 are already active in the GR plane, so one can write (without sending an escape sequence):

```
“English\Français”
```

Alternatively, the Specific Character Set can be specified as:

```
“\ISO 2022 IR 100\ISO 2022 IR 13”
```

In this instance, with the default character repertoire implied by the empty first value, one would have to send an escape sequence<sup>15</sup> before sending the accented character in the second value of the string (i.e. after the first backslash). For example:

```
“English\ ESC 02/13 04/01 Français”
```

The escape sequence can be sent anywhere in the second value prior to the use of the character ‘ç’. It *cannot* be sent in the first value and expected to persist, since the code planes are reset to the default repertoire after every backslash delimiter.

To encode the Katakana characters, one first needs to activate the G1 element of JIS X 201 in the GR plane, by sending the escape sequence “ESC 02/09 04/09” in the third value (before the first Katakana character):

```
“English\Français\ ESC 02/09 04/09 ニホンゴ ”
```

12. The GL plane includes the graphic characters in the low half of the byte, i.e. when the high bit is zero. The GR plane includes the graphic characters in the high half of the byte, i.e. when the high bit is one.

13. Remember though, that not all string value representations in DICOM permit multiple lines.

14. This means that if the default character repertoire is initially active, one can not encode any characters with the high bit set, until an ISO 2022 escape sequence is sent.

Recall that the ISO 8859-1 (ISO-IR 100) G1 plane is reactivated as the default at the beginning of each value, including the beginning of the third value (i.e. after the second backslash). One *cannot* send:

“English\Franç ESC 02/09 04/09 ais¥ ニホンゴ ”

since that would actually be interpreted as:

“English\Français\ÉjézÉiÉS”

The following table lists the available character sets defined for use with ISO 2022 in DICOM, together with the allowed escape sequences.

TABLE 16. Single Byte Character Sets with Code Extensions

	Defined Term	G0 Element	G1 Element	G0 Escape	G1 Escape
Default	ISO 2022 IR 6	IR 6 (ISO 646)	- none -	02/08 04/02	- none -
Latin #1	ISO 2022 IR 100	IR 6 (ISO 646)	IR 100 (ISO 8859-1)	02/08 04/02	02/13 04/01
Latin #2	ISO 2022 IR 101	IR 6 (ISO 646)	IR 101 (ISO 8859-2)	02/08 04/02	02/13 04/02
Latin #3	ISO 2022 IR 109	IR 6 (ISO 646)	IR 109 (ISO 8859-3)	02/08 04/02	02/13 04/03
Latin #4	ISO 2022 IR 110	IR 6 (ISO 646)	IR 110 (ISO 8859-4)	02/08 04/02	02/13 04/04
Cyrillic	ISO 2022 IR 144	IR 6 (ISO 646)	IR 144 (ISO 8859-5)	02/08 04/02	02/13 04/12
Arabic	ISO 2022 IR 127	IR 6 (ISO 646)	IR 127 (ISO 8859-6)	02/08 04/02	02/13 04/07
Greek	ISO 2022 IR 126	IR 6 (ISO 646)	IR 126 (ISO 8859-7)	02/08 04/02	02/13 04/06
Hebrew	ISO 2022 IR 138	IR 6 (ISO 646)	IR 138 (ISO 8859-8)	02/08 04/02	02/13 04/08
Latin #5	ISO 2022 IR 148	IR 6 (ISO 646)	IR 148 (ISO 8859-9)	02/08 04/02	02/13 04/13
Latin #6	-none-				
Thai	ISO 2022 IR 166	IR 6 (ISO 646)	IR 166 (TIS 620)	02/08 04/02	02/13 05/04
Katakana	ISO 2022 IR 13	IR 14 (JIS X 201)	IR 13 (JIS X 201)	02/08 04/10	02/09 04/09

15. By convention, escape sequences are displayed in the same way as characters from a code table, specifically, by their position in the code table, indicated by a “column/row” pair of zero padded decimal values. For an 8-bit code, the column is actually encoded in the high “nibble” (4 bits) and the row in the low nibble. Seven-bit codes will have column values from 0 to 7, 8-bit codes from 0 to 15. For example, in ISO 646, the character ‘A’ would be written as “04/01” and actually encoded as 0x41. The character ‘N’ would be written as “04/14” and actually encoded as 0x4E.

An escape sequence starts with the ESC character, which might be written octal \033, hex 0x1B, or “01/11”. Only the ISO 2022 “code-identification functions” are used in DICOM. This means that the character following the ESC will be “02/08” (G0-designate 94-set), “02/09” (G1-designate 94-set), or “02/13” (G1-designate 96-set). For multi-byte character sets, “02/04” will be followed by either “02/08” (G0-designate multibyte 94-set) or “02/09” (G1-designate multibyte 94-set). The next byte (“final byte”) identifies the set. A special case is “02/04” with a final byte of “04/00,01,02” in which case the second byte “02/08” is omitted (e.g. IR 87).

Shifts and locking shifts are never used in DICOM. The G0 set is always “shifted” into the GL area, and the G1 set is always “shifted” into the GR area.

## Multibyte Character Sets

One more aspect of character sets remains to be discussed. Obviously character sets based on Chinese characters cannot be encoded in a single byte. There are many more characters than the 256 that a single byte could possibly encode. These are referred to as multibyte character sets. DICOM currently has support for Japanese Kanji characters, in addition to Korean Hangul<sup>16</sup> and (a limited set of) Hanja. Up to the present time, there has been no request to add Chinese character sets to DICOM, though the same mechanisms could be used.

There are some interesting differences between the way in which Japanese and Korean multibyte character sets are invoked, not because of any fundamental difference, but because industry practice in each country has led to a preference for invocation in the GL or GR plane, and that preference is followed in DICOM.

Table 17 lists the defined terms and escape sequences for multibyte character sets.

**TABLE 17.** Multibyte Character Sets with Code Extensions

	Defined Term	G0 Element	G1 Element	G0 Escape	G1 Escape
Kanji	ISO 2022 IR 87	IR 87 (JIS X 0208) <sup>a</sup>	- none -	02/04 04/02	- none -
Suppl. Kanji	ISO 2022 IR 159	IR 159 (JIS X 0212)	- none -	02/04 02/08 04/04	- none -
Hangul and Hanja	ISO 2022 IR 149	- none -	IR 149 (KSX 1001)	- none -	02/04 02/09 04/03

a. See "<ftp://ftp.tiu.ac.jp/pub/jis/jisx0208/>".

Notice that the Japanese practice is to invoke the multibyte G0 element in the GL plane, whereas the Korean practice is to invoke the multibyte G1 element in the GR plane. Neither practice defines characters for the other, unassigned, plane. This leaves the user with the responsibility to either:

- ▶ not use the unassigned plane
- ▶ use one of the other, single byte, character sets (by invoking it in the unassigned plane with an appropriate escape sequence).

The key point is that one may have a multibyte character set active in one plane, and a single byte character set in the other. The receiving application must take care to switch based on the high bit of the first byte, and to consume one or more bytes appropriately.

When encoding Korean language strings, the GL plane is usually occupied by ASCII (ISO 646). This is not implicit in the use of the "ISO 2022 IR 149" defined term for

16. Hangul is a constructed phonetic character set, but there more characters defined than 256 so it also needs a multibyte character set. Hanja are Chinese characters used with the Korean language.

Specific Character Set. Therefore one also has to specify either a null value to invoke the default repertoire in GL, or ISO 2022 IR 6 specifically. Usually the first value is left null, the second value is “ISO 2022 IR 149”, and “ESC 02/04 02/09 04/03” is sent before each Hangul or Hanja text value.

When encoding Japanese language strings, it is more common to invoke a character set in GL that includes Kanji, Katakana and Hiragana by specifying “ISO 2022 IR 87”. Either the default character repertoire or Romaji from JIS X 201 invoked by “ISO 2022 IR 13”<sup>17</sup> is also used in GL, with the appropriate escape sequences to alternate between them. GR is completely ignored. When Romaji is required, the appropriate escape sequence is used to activate the JIS X 201 G0 element in the GL plane to temporarily replace the Kanji characters, without requiring a third value of Specific Character Set.

An interesting example of using a mixture of single and multibyte characters in DICOM is the person name (PN) value representation. Recall from the chapter on “Content Items: Concept Names and Values” that more than one “component group” can optionally be provided for a name. These component groups are a single-byte (usually English) representation, an ideographic representation and a phonetic<sup>18</sup> representation separated by an ‘=’ (0x3D) character.

Here is an example of a Japanese name with component groups:

```
“Miyamoto^Musashi= 宮本^武蔵 = みやもと ^むさし ”
```

Notice that one still uses the ‘^’ symbol to delimit components of the name (first name, family name and so on) within the phonetic and ideographic representations. However, the ‘^’ may well need to be represented using a single byte Latin or Romaji character set invoked in GL, necessitating a few extra escape sequences in amongst the phonetic and ideographic representations.

Given a Specific Character Set of “\ISO 2022 IR 87”, with the escape sequences visible, the same string looks like:<sup>19</sup>

```
“Miyamoto^Musashi=
ESC 02/04 04/02 宮本
ESC 02/08 04/02 ^
ESC 02/04 04/02 武蔵
ESC 02/08 04/02 =
ESC 02/04 04/02 みやもと
```

17. The use of IR 13 here is a little confusing, since the registry number for the Romaji part of JIS X 201 is IR 14 not IR 13. The Katakana part of JIS X 201 is IR 13. However in DICOM, the defined term “ISO 2022 IR 13” invokes the Romaji in GL and the Katakana in GR at the same time.

18. In Japan, there are two phonetic writing systems: Hiragana and Katakana. Hiragana is used for native Japanese words (including people’s names) and Katakana is used primarily for words of foreign origin. While there is a single byte Katakana character set, there is no single byte Hiragana character set in common use or supported in DICOM, and so IR 87 must be used.

19. The string is split into multiple lines for readability; obviously there are no new lines in the actual name string when it is encoded.

```
ESC 02/08 04/02 ^
ESC 02/04 04/02 むさし”
```

Given a Specific Character Set of “ISO 2022 IR 13\ISO 2022 IR 87”, with the escape sequences visible, the same string looks like:

```
“Miyamoto^Musashi=
ESC 02/04 04/02 宮本
ESC 02/08 04/10 ^
ESC 02/04 04/02 武蔵
ESC 02/08 04/10 =
ESC 02/04 04/02 みやもと
ESC 02/08 04/10 ^
ESC 02/04 04/02 むさし”
```

In both examples, notice that:

- ▶ the character set of the first value of Specific Character Set is already activated at the start of the text value, and doesn’t need to be explicitly invoked
- ▶ all of the delimiters are “escaped” into a single byte representation since DICOM delimiters are only defined for single byte character sets

The examples so far have used “anglicized” names in the first component group. It is possible to use any single byte character set representation in the first component group. Accordingly, one could use the Katakana in IR 13 together with the multiple Kanji and Hiragana representations using a Specific Character Set of “ISO 2022 IR 13\ISO 2022 IR 87” as follows:

```
“ミヤモト^ムサシ=
ESC 02/04 04/02 宮本
ESC 02/08 04/10 ^
ESC 02/04 04/02 武蔵
ESC 02/08 04/10 =
ESC 02/04 04/02 みやもと
ESC 02/08 04/10 ^
ESC 02/04 04/02 むさし”
```

Again, notice how the first value of Specific Character Set, in this case IR 13, is already activated at the start of the text value, and doesn’t need to be explicitly invoked.

Like Japanese, the Korean language also has the concept of phonetic and ideographic representations. Accordingly, component groups are also used for Korean names. The difference with Korean is that it is common practice to invoke IR 149 and use the multibyte Hangul and Hanja characters in GR, while at the same time leaving the default (single byte) character repertoire (ASCII) active in GL. Whether to use one or two bytes for each character is determined by the high bit of the first byte.

For example, given a Specific Character Set of “\ISO 2022 IR 149”<sup>20</sup> one can encode:

20. It would be unusual to see any other values than these for Specific Character Set when encoding Korean.

```
“Lim^Seong-min=  
ESC 02/04 02/09 04/03 林  
^  
ESC 02/04 02/09 04/03 成敏  
=  
ESC 02/04 02/09 04/03 임  
^  
ESC 02/04 02/09 04/03 성민 ”
```

Notice in this example that:

- ▶ the default repertoire is always active in GL, so delimiters do not need to be escaped
- ▶ the IR 149 G1 element is activated in GR after each delimiter (by an escape sequence), since DICOM specifies that there is a return to the initial state (GR undefined in this case) after each delimiter (which includes ‘^’ and ‘=’ as well as ‘\’)

## Why not Unicode?

Why is all this complexity necessary? Why not just use Unicode? After all, Unicode is supposed to unify all these annoying and inconsistent character sets from all countries, allow one to internally make use of “wide characters” of fixed length sufficient to encode anything, and provide standard methods of serializing the internal fixed width characters to external variable length characters. However:

- ▶ most locales<sup>21</sup> that needed multibyte characters (before Unicode) already have a huge installed base of systems that use “old-fashioned” localized character sets<sup>22</sup>
- ▶ most locales that use single byte character sets beyond ASCII, already have a huge installed base of systems that use the ISO 8859 family (or other similar nationally specific character sets<sup>23</sup>)
- ▶ the “Han Unification” which folded similar Chinese characters into single entries in Unicode regardless of whether they were used in China, Korea or Japan, apparently causes problems of interpretation and rendering

Accordingly, despite the enormous investment of time and energy in developing Unicode, adoption in some locales is slow. Partly this is because determining which encoding to use for external representation of strings is only part of a localization effort. Other (more difficult) aspects of localization include translation of concepts into different languages, support of appropriate fonts, localized “input methods,”<sup>24</sup> localized keyboards, and so on.

---

21. “Locale” is just a fancy word for “place” (especially with reference to a particular event). The word is consistently used in the literature on internationalization and localization.

22. This is especially true in Japan. Unicode does not appear to be very popular in Japan.

23. For example the Thai TIS 620 character set.

24. An “input method” is a way of getting funky characters out of a conventional keyboard. For example, several keystrokes may be compounded to bring up a pick-list of potential ideographs.

The localization effort is simplified by the existence of various free and commercial-off-the-shelf tools and libraries for translating between representations, whether one uses Unicode or not. Java, which uses Unicode internally for all strings, provides a plethora of converters for common encoding schemes to use when reading and writing strings.<sup>25</sup> XML also supports Unicode. Indeed all XML parsers are required to support both UTF-8 and UTF-16.<sup>26</sup>

The DICOM Committee was at one point asked to consider adding various Unicode encodings (such as UTF-8 and UTF-16) to the list of defined terms supported for Specific Character Set. The rationale was that implementers could then choose to use UTF-8 or UTF-16 rather than ISO-IR 87 to represent Kanji. In addition, since Unicode already incorporates support for many countries not yet supported in DICOM, one wouldn't need to keep adding nationally specific solutions.<sup>27</sup> The proposal was rejected on the grounds that there would then be two incompatible ways to do the same thing, equipment would not interoperate and objects could not be interchanged.

## Character Sets and Conformance

Earlier it was described how, when an association is established, two devices exchange information about their capabilities. Specifically, before objects can be exchanged, both devices have to agree to support the same types of object (SOP classes) and the same encoding or compression scheme (transfer syntax).

This negotiation of capabilities does *not* include negotiation of which character sets are supported. This is potentially a major problem.

For example, the sender (SCU) may transmit a structured report using ISO-IR 87 with Kanji characters in strings like names and code meanings and plain text. The receiver (SCP) may or may not be capable of storing or rendering such strings.<sup>28</sup> Strings encoded in unsupported character sets may be truncated, displayed as garbage, cause value representation violation warnings and errors, be mis-interpreted as control characters, and so on. Furthermore, though reports may be successfully stored in an archive, not all viewing applications will be able to render strings in the more exotic character sets.

Implementations are required to document in their Conformance Statements the extent of their support for various character sets. For devices marketed only in the US, often only the default character repertoire (US-ASCII) is supported. For devices

---

25. Although not always in a precisely standard fashion. For example, UTF-8 in Java is apparently not quite the same as everybody else's UTF-8.

26. US-ASCII is a degenerate subset of UTF-8, with the high bit not set.

27. As has been done for Korean and Thai, for example.

28. In the worst case, if the SCP supports only the default character repertoire (or any single byte character set without code extensions), it may assume that one character always occupies one byte. It may allow buffers containing values whose maximum lengths are specified in characters rather than bytes to overflow, and crash or otherwise behave badly.

sold in Europe, ISO-IR 100 is almost always supported, primarily so that accents in peoples' names are represented correctly.

From the perspective of storing objects across the network, it is unfortunate that character sets cannot be negotiated, since the sender might have the capability to convert strings and re-encode the data using a character set that the receiver was capable of supporting. For interchange media, obviously one must agree in advance on what character sets to support.

The choice of character set also has implications for other DICOM services than storage, particularly the query-retrieve service, and the modality worklist SOP class. These are discussed in the chapter on "Document Management: SR in the Real World." Matching strings during queries, especially matching peoples' names, is hard enough at the best of times. One also has to consider the character set in which:

- ▶ the query is encoded
- ▶ the response is encoded (must it always be the same as the query?)
- ▶ how strings are internally represented within the SCP

For example, will "français" in the default character repertoire match "français" (with a cedilla on the c) in ISO-IR 100 (ISO 8859-1)?

## **Internationalization of Codes**

In earlier chapters, it was mentioned that coded entries also present opportunities for international harmonization. Admittedly, there is some regional variation in the understanding of similar concepts. However most of the variation is caused not by different interpretations, but by different string representations of the same concept.

The meaning of a code may be determined in a number of ways. "Meaning" is used here in the sense of an appropriate string value to use for presentation in a particular locale.<sup>29</sup> One might:

- ▶ use the transmitted code meaning string (which is always present in DICOM)
- ▶ look up the code in a dictionary
- ▶ examine the "concept modifiers"<sup>30</sup> for an appropriate text meaning

Consider the example of trying to express the meaning of the concept of a "finding" (perhaps to use as a heading in a structured report). In an English-speaking locale one might simply encode:

```
(0008,0100) Code Value "209001"  
(0008,0102) Coding Scheme Designator "99PMP"  
(0008,0104) Code Meaning "Finding"
```

---

29. As opposed to an actual definition of the concept that the code represents.

30. That is, the target nodes of "has concept modifier" relationships, which modify, qualify or replace the meaning of the concept name code sequence.

Remember that the Code Meaning attribute is mandatory, and should contain a string from the coding scheme from which the code was chosen. One should not just substitute one's own (localized) meaning. Here is the same triplet code abbreviated as a tuple in the usual manner:

(209001,99PMP,“Finding”)

As an implementation short-cut, this particular code meaning could be rendered directly by a device in an English-speaking locale. However, an English dictionary should still probably be used to extract an appropriate meaning to render. A system that takes short cuts like this will be out of luck if it later receives a coded entry with a code meaning in another language. If the system always renders the transmitted meaning literally, it will appear in the transmitted language, not English as expected. For example, when receiving:

(209001,99PMP,“Befund”)

or:

(209001,99PMP,“所見”)

a directly rendered meaning will not make a lot of sense to a reader who can't understand German or Japanese. As an additional complication, for the meaning to be renderable at all requires that the appropriate character set be supported, and the necessary fonts be available. Otherwise the meaning may be rendered as garbage.

Had the same system looked up the code in its own local dictionary populated with English meanings, then a meaning could have been rendered that was understandable to the reader.

Notice that there is no way of signaling the intended locale within the coded entry. The receiver cannot determine in what language the code meaning field is encoded,<sup>31</sup> without some mechanism outside the coded entry itself. This problem will be discussed in more detail in a later section.

Whether or not one can send localized strings in the code meaning field is a little controversial. The goal is to avoid implementors “making up” their own synonyms, so the standard specifies that the code meaning must come from the coding scheme. This does not mean it must be in English. Unfortunately, there is a paucity of code sets with multilingual meanings for the same concepts. In particular, there are few, if any, translations of the coding schemes commonly used in DICOM to languages other than English. The solution is clearly to make such translations, then incorporate them into schemes like SNOMED. DICOM has negotiated the right to do this work. It is hoped that the benefit of being able to create common codes for use in different regions will spur implementers and users to develop multilingual code sets, at least for more commonly used terms.

---

31. Obviously, the character set used may give some hint, but many different languages can be encoded using the same character set, and different character sets can be used in the same string with code extension techniques (ISO 2022 escape sequences).

Encouraging implementers to develop multilingual coding schemes, and to depend on dictionaries (rather than transmitted code meaning strings) allows for the possibility of:

- ▶ deploying the same applications in multiple international markets
- ▶ interchanging reports across national language boundaries

Even for a single vendor, to be able to localize the same products for sale in different countries, producing or contributing to translations of coding schemes is worthwhile.

Regardless, it is clearly unacceptable in regions where the primary language is not English to expect everyone to encode English strings. Indeed, in some countries it is probably illegal to insist on displaying one language alone (especially when that language is English).

In summary then, the recommended strategy for general use of coded entries is:

- ▶ as a receiver, to always look up the code value and coding scheme designator pair in a localized lexicon, use the localized meaning from the lexicon, and fall back on the transmitted code meaning only if the look-up fails
- ▶ as a sender, to always send a localized code meaning when it is specified in the coding scheme, just in case the receiver does not have the same dictionary; when there is no localized meaning available, an English code meaning should be sent<sup>32</sup>

## Concept Modifiers and Languages

There is another approach that is applicable to concept names of content items (and also indirectly to the use of codes as values and units): to make use of the “has concept modifier” relationship. As has already been described, this relationship was introduced in order to:

- ▶ support code meanings longer than 64 characters
- ▶ construct post-coordinated concepts
- ▶ support multiple free text language translations

In the context of internationalization and localization, the “has concept modifier” relationship can be used to provide multiple equivalent language translations of the same concept. Though the codes and templates to do this have not yet been standardized, the basic principles are described in the remainder of this section.

The original idea was to do some like this:

```
<CONTAINER:(209001,99PMP,“Finding”)>
  <has concept mod TEXT:
    (209601,99PMP,“German meaning of value”)=“Befund”>
  <has concept mod TEXT:
```

---

32. It is pretty unlikely that a non-English system is going to generate codes that don't have an appropriately localized meaning. However, the localized meaning might be locally produced, and not be authorized by the coding scheme standard, in which case it is not supposed to be sent.



(209602,99PMP,"Japanese meaning of value")="所見">

This approach is analogous to the pattern for which the relationship was originally conceived, to transmit meanings longer than 64 characters:

```
<CODE:(209007,99PMP,"Procedure")=(50291CC,I10P,
  "IMAGING:CNS:CT:SELLA:LOWOSMOLAR:IT,U,E:2PLANE3D")
<has concept mod TEXT:
  (209600,99PMP,"Equivalent meaning of value")=
  "CT imaging study of the sella turcica/pituitary
  gland, unenhanced, with intravenous contrast, and
  with intrathecal contrast, using low osmolar contrast,
  in two planes, with 3D reconstructions">
```

Unfortunately, this approach is not sufficient for a number of reasons:

- ▶ identifying a “language” is not as easy as it sounds; there may be more than one language per country, and the same language may be used (slightly differently) in multiple countries; accordingly, language definitions may need to be post-coordinated
- ▶ producing pre-coordinated codes to represent “meaning in language xx” is an onerous task; it involves shadowing all the codes for “language xx,” or worse, “language xx in country yy,” not to mention keeping up to date with changes in language and country definitions
- ▶ it is necessary to indicate in which language values themselves (not just concept names) are encoded
- ▶ it is necessary to provide alternative equivalent meanings in different languages for both values and concept names

## Identifying Languages

There are multiple dimensions to representing text in a particular language. These include defining:

- ▶ the language itself
- ▶ the country in which it is being used
- ▶ the character set used to encode the string
- ▶ the fonts used to display or print the string

The font issues are beyond the scope of DICOM Structured Reporting.<sup>33</sup> The character set mechanisms have already been described at length.

The convention for identifying a language by the combination of a language code and a country code is a well established information technology practice (widely used on the Internet).<sup>34</sup>

33. Though fonts are interchanged in some standards that are more presentation oriented. For example, both Adobe Postscript and PDF support font embedding, which means that a document can be rendered anywhere (whether it can be understood or not).

Codes for languages are defined in ISO 639-1, and consist of two letters for each language. ISO DIS<sup>35</sup> 639-2 defines three letter codes to allow for a larger repertoire of languages. One needs three letters to specify the language of a report written in Aztec for example.<sup>36</sup> RFC 1766 also specifies a few more unusual languages beyond the ISO standard.<sup>37</sup> The codes for countries that are used to qualify the language are drawn from the two-letter codes in ISO 3166-1.

What is proposed for DICOM is to define a template that allows the language to be specified in a manner consistent with Internet practice. This involves using content items and relationships to construct a post-coordinated description assembled from ISO 639 and ISO 3166 codes.

For example, one could try to specify the language of a concept or value by using something like:

```
<TEXT:(209104,99PMP,"Anatomie")="Sein">
  <has properties CODE:(209650,99PMP,"Language")=
    (fr,IS0639_1,"Français")>
  <has properties CODE:(209651,99PMP,"Country")=
    (CA,IS03166_1,"Canada")>
```

The problem with this simplistic approach, is that to be consistent with the normal interpretation of the “has properties” relationship, what is really being said here is that “the anatomy is the breast, which has a language of French and a country of Canada.” While there may indeed be a few French-speaking breasts out there, this construct is clearly ambiguous, not to mention counter to the intent of the “has properties” relationship.

That is not to say that one can’t construct post-coordinated designators of language in general, perhaps by using a container:

```
<CONTAINER:(209650,99PMP,"Language")>
  <has properties CODE:(209650,99PMP,"Language")=
    (fr,IS0639_1,"Français")>
  <has properties CODE:(209651,99PMP,"Country")=
    (CA,IS03166_1,"Canada")>
```

Here the use of “has properties” is kosher, since one is describing an attribute of the container itself.

It might be better and simpler to use the language itself as a primary designator, then qualify that:

```
<CODE:(209650,99PMP,"Language")=(fr,IS0639_1,"Français")>
```

---

34. Specifically, it is described in RFC 1766, “Tags for the Identification of Languages.” For example, US English (as opposed to real English) is designated “en-US”, French Canadian is designated “fr-CA”, and so on. It is not required in the RFC that the “secondary tag” that designates the country be present, since in some cases “en” or “fr” alone might suffice, for example.

35. DIS stands for “Draft International Standard,” that is, one that is not quite done yet.

36. Though not Sanskrit; there is a two letter code for that!

37. Including Klingon, believe it or not.

```
<has properties CODE:(209651,99PMP,"Country")=
      (CA,IS03166_1,"Canada")>
```

Unfortunately, the simpler SR SOP classes, both Basic Text and Enhanced, do not permit the use of code value types as children of other code value types with relationships other than “has concept modifier.” Specifically, though any type of value can have a “has concept modifier” relationship with a text or code value type, only the child text value type can have another child other than by another “has concept modifier” relationship. In essence, code value types must be leaves, except via “has concept modifier” relationships. These restrictions do not apply to the Comprehensive SR SOP class, but ideally one would like to have the same mechanism to specify languages used for all SOP classes.

This means that one has to use the “has concept modifier” relationship rather than the “has properties” relationship:

```
<CODE:(209650,99PMP,"Language")=(fr,IS0639_1,"Français")>
  <has concept mod CODE:(209651,99PMP,"Country")=
      (CA,IS03166_1,"Canada")>
```

In the qualified case, it remains to be determined whether the concept name of the modifier should be something generic like “country,” or something more specific, like “country of language.”

Even in the case of English, outside the US it is probably desirable to define that one is using meanings that are spelled “correctly”:

```
<CODE:(209650,99PMP,"Language")=(en,IS0639_1,"English")>
  <has concept mod CODE:(209651,99PMP,"Country")=
      (GB,IS03166_1,"United Kingdom")>
```

or

```
<CODE:(209650,99PMP,"Language")=(en,IS0639_1,"English")>
  <has concept mod CODE:(209651,99PMP,"Country")=
      (AU,IS03166_1,"Australia")>
```

There is of course a common degenerate case when no qualifier is actually needed, such as in:

```
<CODE:(209650,99PMP,"Language")=(fr,IS0639_1,"Français")>
```

or

```
<CODE:(209650,99PMP,"Language")=(en,IS0639_1,"English")>
```

## Designating Language of Names and Values

Given that a mechanism exists to encode a choice of language, how does one attach the definition to other parts of the SR tree to designate the language used for a name or value? There are two main approaches.

First, at (or close to) the document root, one can designate some aspect of observation context as specifying the language in use, such that it is then inherited by all descendants.

For example, at the root node, one can specify:

```
<CONTAINER:(209076,99PMP,"Chest X-Ray Report")>
  <has obs context CODE:(209652,99PMP,
    "Language of content item and descendants")=
    (en,IS0639_1,"English")>
```

Alternatively, one can use such a mechanism to designate two parallel sub-trees that are equivalent translations in different languages.<sup>38</sup> For example:

```
<CONTAINER:(209076,99PMP,"Chest X-Ray Report")>
  <CONTAINER:>
    <has obs context CODE:(209652,99PMP,
      "Language of content item and descendants")=
      (en,IS0639_1,"English")>
    ... english language report goes here ...
  <CONTAINER:>
    <has obs context CODE:(209652,99PMP,
      "Language of content item and descendants")=
      (fr,IS0639_1,"Français")>
    ... french language report goes here ...
```

At a more atomic level, one can define the language of individual content items. For example:

```
<TEXT:(209104,99PMP,"Anatomy")="Sein">
  <has concept mod CODE:(209654,99PMP,
    "Language of value")=
    (fr,IS0639_1,"Français")>
  <has concept mod CODE:(209651,99PMP,"Country")=
    (CA,IS03166_1,"Canada")>
```

Strictly speaking, the “has concept modifier” relationship modifies only the meaning of the concept itself.<sup>39</sup> Therefore, in order to specify the language of the *value* rather than just the name of the name-value pair, modifiers are needed that explicitly designate the:

- ▶ “language of the name”
- ▶ “language of the value”
- ▶ “language of the name and value”

Compare the following examples two with the last one:

```
<TEXT:(209104,99PMP,"Anatomie")="Sein">
  <has concept mod CODE:(209654,99PMP,
    "Language of name and value")=
```

---

38. It would also be desirable to indicate somehow that these two sub-trees were parallel and equivalent.

39. Indeed, the examples in Supplement 23 in the definition of the “has concept modifier” relationship are flawed, since they do not distinguish the code that is the concept name from the code that is the value.

```

                                (fr,IS0639_1,"Français")>
<has concept mod CODE:(209651,99PMP,"Country")=
                                (CA,IS03166_1,"Canada")>

<TEXT:(209104,99PMP,"Anatomie")="Breast">
  <has concept mod CODE:(209655,99PMP,
    "Language of name")=
                                (fr,IS0639_1,"Français")>
  <has concept mod CODE:(209651,99PMP,"Country")=
                                (CA,IS03166_1,"Canada")>

```

## Equivalent Meanings

The mechanisms discussed so far only describe the language used to encode a single meaning of a concept name or value. One may also need to convey alternative or equivalent meanings in different languages.

There are several possible approaches. For the concept name itself, one can encode:<sup>40</sup>

```

<CONTAINER:(209001,99PMP,"Finding")>
  <has concept mod TEXT:(209599,99PMP,
    "Equivalent meaning of name")="Befund">
  <has concept mod CODE:(209654,99PMP,
    "Language of value")=(de,IS0639_1,"Deutsch")>

```

The meaning of the same concept may then be expressed in several alternative languages:

```

<CONTAINER:(209001,99PMP,"Finding")>
  <has concept mod TEXT:(209599,99PMP,
    "Equivalent meaning of name")="Befund">
  <has concept mod CODE:(209654,99PMP,
    "Language of value")=(de,IS0639_1,"Deutsch")>
  <has concept mod TEXT:(209599,99PMP,
    "Equivalent meaning of name")="所見">
  <has concept mod CODE:(209654,99PMP,
    "Language of value")=(ja,IS0639_1,"Nihongo")>

```

Notice that with this construction, the fact that the alternative meanings of the concepts are in different languages is conveyed in a post-coordinated manner. Specifically, one doesn't know what language has been used for a particular meaning until one examines the target of its "has concept modifier" relationship.

Also notice that with this approach there is always a "primary" meaning, contained in the code meaning attribute of the concept name code sequence that is being qualified with additional meanings. This does not have to be an English meaning. The coding

40. The "has concept modifier" relationship is again used instead of the "has properties" relationship because of the SOP class limitations described in the earlier section, in case the language needs to be further qualified by country.

scheme may also define, for example, a Japanese meaning for the code for “finding,” in which case one can encode it directly as the primary meaning:

```
<CONTAINER:(209001,99PMP,“所見”)>
  <has concept mod TEXT:(209599,99PMP,
    “Equivalent meaning of name”)=“Befund”>
    <has concept mod CODE:(209654,99PMP,
      “Language of value”)=(de,IS0639_1,“Deutsch”)>
  <has concept mod TEXT:(209599,99PMP,
    “Equivalent meaning of name”)=“Finding”>
    <has properties CODE:(209654,99PMP,
      “Language of value”)=(en,IS0639_1,“English”)>
```

The language of the “primary” meaning is not specified, unless it is explicitly conveyed as a modifier.<sup>41</sup> For example:

```
<CONTAINER:(209001,99PMP,“所見”)>
  <has concept mod CODE:(209655,99PMP,
    “Language of name”)=(ja,IS0639_1,“Nihongo”)>
  <has concept mod TEXT:(209599,99PMP,
    “Equivalent meaning of name”)=“Befund”>
    <has concept mod CODE:(209654,99PMP,
      “Language of value”)=(de,IS0639_1,“Deutsch”)>
```

It is also necessary to be able to specify alternative meanings in different languages for not only the concept name, but also the value. The “has concept modifier” relationship only applies to the concept. However, it is (probably) valid to use a specific modifier concept name to indicate that the modifier applies to the value, rather than the name, of the content item being modified. For example:

```
<CODE:(209104,99PMP,“Anatomy”)=(T-04000,SNM3,“Breast”)>
  <has concept mod TEXT:(209599,99PMP,
    “Equivalent meaning of name”)=“Anatomie”>
    <has concept mod CODE:(209654,99PMP,
      “Language of value”)=(fr,IS0639_1,“Français”)>
  <has concept mod TEXT:(209600,99PMP,
    “Equivalent meaning of value”)=“Sein”>
    <has concept mod CODE:(209654,99PMP,
      “Language of value”)=(fr,IS0639_1,“Français”)>
```

Note that one cannot specify alternative meanings for both the name and the value in the same content item, since one needs two text strings to achieve this. Each of the modifiers also needs to be qualified by its own language designator, which applies to the value of the parent modifier (not its name).

In all of the examples shown so far, the alternative language meaning has been encoded as plain text in a TEXT value type. This does not have to be the case. If an appropriate code is available it can be used, whether it be:

41. Or unless there is a language specified higher up in the tree that applies generally, as discussed before.

- ▶ the same code value and coding scheme designator with an alternative standard code meaning
- ▶ a different code value, from the same or a different coding scheme

For example, if the same concept name is defined by different codes in separate English and French coding schemes, one could write:

```
<CONTAINER:(CV1,CSD1,"Anatomy")>  
  <has concept mod CODE:(209599,99PMP,  
    "Equivalent meaning of name")=(CV2,CSD2,"Anatomie")>
```

Notice the use of the CODE rather than TEXT value type. One could elaborate further and specify the language of the synonyms, for both the "primary" concept name and the equivalent meaning:

```
<CONTAINER:(CV1,CSD1,"Anatomy")>  
  <has concept mod CODE:(209655,99PMP,  
    "Language of name")=(en,IS0639_1,"English")>  
  <has concept mod CODE:(209599,99PMP,  
    "Equivalent meaning of name")=(CV2,CSD2,"Anatomie")>  
  <has concept mod CODE:(209654,99PMP,  
    "Language of value")=(fr,IS0639_1,"Français")>
```

## Reality Check - Codes and Languages in Practice

The issues discussed in this chapter are complex, though the requirements described are real. "Provincial" implementations may choose to ignore them at their own risk, and forego participation in international markets.

Fortunately, the character set support in DICOM is well-established, if not actually widely deployed. Various solutions are proposed for the support and identification of languages in structured reports, but these are likely to be subject to considerable revision before any final standard is adopted.

The most important lessons of this chapter are:

- ▶ to design implementations that can accept codes from different coding schemes
- ▶ to use the meaning of a code from an appropriate localized dictionary (preferably one which is configurable at run time)
- ▶ not to depend on the code meaning field transmitted in the data

Always try to avoid rendering the code meaning field literally, unless the code value isn't found in the dictionary, or there are serious issues with rendering synonyms.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ DICOM does not support only US or English language environments
- ▶ the extensive use of text, codes and peoples' names in structured reports raises the priority of the issue of languages and character sets

- ▶ the “default character repertoire” is 7-bit US-ASCII (aka. ISO 646 or ISO-IR 6)
- ▶ replacements for the default character repertoire are listed in the attribute Specific Character Set
- ▶ a single Specific Character Set value replaces the default character repertoire with the specified single byte character set
- ▶ the most commonly supported single byte character set is ISO 8859-1 (aka. ISO-IR 100 or Latin Alphabet No. 1), which includes support for most Western European accented characters
- ▶ single byte character sets that suffice for most countries are already defined in DICOM; multibyte character sets are defined for Japan and Korea
- ▶ additional country-specific character sets can be added to DICOM on request
- ▶ multiple character sets can be used in the same string, by using the *very limited subset* of ISO 2022 escape sequences that are defined in DICOM for code extension
- ▶ multibyte character sets are also activated using code extension techniques (escape sequences)
- ▶ Unicode is not supported in DICOM objects, and likely will never be; that is no reason not to use it inside an application if appropriate, and convert strings on input and output for interchange
- ▶ support for character sets is not negotiated on the network, but is defined in the conformance statement
- ▶ the use of different character set encodings during queries may be problematic
- ▶ coded entries in structured reports create new challenges for character sets and languages
- ▶ the code meaning field of a coded entry (triplet code) may only contain a meaning defined in the coding scheme, not an arbitrary (non-standard) localized equivalent
- ▶ the code value and coding scheme designator should be looked up in a dictionary to find an appropriate, localized meaning, rather than rendering the transmitted code meaning literally
- ▶ the “has concept modifier” relationship can be used to provide one or more translations of the meanings of concept names, and potentially of the meanings of content item values as well
- ▶ it is possible to encode a single report in multiple languages, either by the exclusive use of codes that can index a local dictionary, or by the provision of text or code meanings in multiple languages
- ▶ languages are specified by codes, potentially modified by country; well recognized standards exist for specifying both
- ▶ the language of a name or value can be described either by using a language specified in observation context and inherited, or attached to each content item
- ▶ equivalent meanings in different languages can be defined for both concept names and values; these can be specified as plain text or as codes (if appropriate localized code meanings are available)



# ***Security: Privacy and Authenticity***

---

When DICOM was concerned only with the transfer of images, mostly within a single organization, matters of security were of little concern. As enterprises have grown and become distributed, more users connect from remote sites. Remote connections have also become faster and cheaper. Indeed, public networks like the Internet have become sufficiently cheap and reliable that they may be used for the exchange of health-care information, including reports and images.

SR documents, whether they contain traditional reports about diagnostic procedures or other kinds of structured information, are even more likely to be exchanged over public networks than images. Accordingly, matters of security are relevant to any discussion of SR.

A little knowledge of security is a dangerous thing,<sup>1</sup> so wherever possible DICOM has tried to make use of existing approaches from other industries (and common Internet practice), rather than going its own way or reinventing the wheel. The author freely admits to being a rank amateur on the subject, so this chapter should not be regarded as gospel, far from it. It is only included to summarize the issues involved, and highlight the potential solutions that are available in DICOM. *Always consult some one proficient in the subject of security before implementing or deploying a security solution.*

## **What is "Security?"**

There are many different aspects to securing information in an organization. Most of them involve implementing a comprehensive security policy and enforcing it through

---

1. However, if one must gain a little knowledge about the subject, Bruce Schneier's book is strongly recommended. There are other good books, but none is as readable, gives as comprehensive an overview, nor is any as entertainingly irreverent. "Schneier D. *Applied Internet Security*. Wiley. ISBN 0-471-11709-9."

training and constant vigilance. No amount of technology can compensate for inadequate physical security or inadequate procedures. An unlocked door, an overheard conversation, a document in a trash can, or a password on a sticky note on a monitor will compromise the most secure technology based security solution.

Having said that, if one can assume an appropriate environment on either end of a network connection or media transfer, what can be done from a DICOM perspective to “secure” information? The possibilities include:

- ▶ confirming the identity of the parties involved, that is “authenticating” that they really are who they claim to be
- ▶ restricting access to specific information on the basis of the identity of an individual (or where they are located), that is, “access control”
- ▶ recording who has accessed what and when, either appropriately or inappropriately, that is, maintaining an “audit trail”
- ▶ protecting information from interception or eavesdropping while it is in transit, that is, establishing “confidentiality”
- ▶ recording that a document or object has indeed been created by a particular individual, that is, assigning it a “digital signature”
- ▶ verifying that a document or object has not been altered since it was created, that is, verifying its “integrity”
- ▶ recording that a document was actually created (and perhaps when), in a manner that can’t subsequently be refuted, that is, making it “non-repudiable”

These possibilities are not mutually exclusive and there is considerable overlap. The same technology can be used to provide some or all of these features. Generally, some sort of cryptographic algorithms are involved. Indeed it is largely the widespread availability of reliable, low-cost cryptography that has spurred greater interest in security in the civilian sector.

To make sure that one doesn’t get involved in deploying complex and potentially inconvenient security solutions, it is vital to establish requirements first. In particular, one must consider local, national and international regulatory requirements and limitations. Having incorporated statutory requirements, one must then analyze the level of threat to the information. For example, it is fair to say that a patient’s CT images are at less risk than their credit card number. Despite the low level of risk to information about them, patients value their privacy (and rightly so). Accordingly, straightforward measures to protect patient data may be worthwhile. Nowadays, with the widespread availability of solutions created to meet the (more stringent) demands of electronic commerce, there is little excuse for not implementing at least routine security measures.

## **DICOM and Security**

DICOM has restricted its involvement in security to only those aspects directly related to the transfer and encoding of objects. Specifically DICOM supports, or will support:

- ▶ peer authentication, message integrity and confidentiality over the network using TLS (aka SSL)<sup>2</sup>
- ▶ digital signatures for authentication and integrity of individual objects<sup>3</sup>
- ▶ confidentiality of objects stored on interchange media<sup>4</sup>
- ▶ confidentiality of individual attributes on media and over the network<sup>5</sup>

Other aspects of security are considered outside the scope of the standard. For example, there is no means of directly providing access control in DICOM. There is no “login” procedure for example, nor any mechanism to exchange information directly about who is accessing the data. However, as will be shown, the TLS/SSL mechanism does provide the necessary tools with which to restrict access, as well as to maintain audit trails, in addition to providing authentication, integrity and confidentiality.

## TLS/SSL

A book on structured reporting is not the place to go into great detail about the TLS protocol.<sup>6</sup> A summary of the features of TLS include the ability to:

- ▶ confirm the identity of the “server,” i.e. the device to which the connection is being made, by verifying that they are in possession of a “digital certificate” known to belong to them
- ▶ optionally, have the server verify the identity of the “client” in the same manner
- ▶ establish a cryptographically protected connection such that eavesdropping is thwarted
- ▶ confirm that each packet of information transmitted has not been altered

In other words, TLS supports mutual peer authentication, confidentiality and message integrity.

The key<sup>7</sup> to all this is the use of “digital certificates” based on public key cryptography. The idea of public key cryptography is that a “key” is composed of two halves, one

---

2. *TLS is Transport Layer Security, and is the Internet Engineering Task force (IETF) standard version of what Netscape developed as Security Sockets Layer (SSL). TLS 1.0 as specified in DICOM is based on SSL 3.0, the most current version in use on the Internet for electronic commerce.*

*DICOM also supports another network security option, ISCL, which was developed in Japan and is designed for use with smart cards.*

3. *Digital signatures are proposed in Supplement 41 which has not yet gone to ballot.*

4. *Media security is the subject of Supplement 51, which was in early draft form when this was written.*

5. *Attribute level confidentiality is the subject of Supplement 55, also in early draft form. The ability to selectively protect individual attributes may be useful in scenarios in which some individuals are permitted access to different sets of information than others.*

6. *An excellent book on the subject is “Thomas S. SSL and TLS Essentials. Wiley, ISBN 0-471-38354-6.” It is a good thing it is excellent, because it is also currently the only book on the subject.*

7. *If you will excuse the pun.*

publicly known and associated with the owner of the key, and the other private, known only to the owner, and closely guarded. Anyone with the public key can create a message that only the possessor of the private key<sup>8</sup> can read. The possessor of the private key can sign messages that anyone with the public key can verify as having been signed by that person.

## **TLS and Authentication of Individual Identity**

In the case of TLS, the client can confirm that the server is who they claim to be by sending them a message encrypted with the server's public key, that only the possessor of the private key (which should be the server) can decipher.

However, since there may not be an infrastructure in place to distribute everyone's digital certificates with public keys, an additional trick is needed. In TLS, the server sends the public key to the client in the first place, so obviously they are likely to have the corresponding private key. How does the client know the server is who it claims to be? The answer is that the certificate is signed by someone else, someone who the client trusts, someone who verifies that the certificate belongs to whom it claims to belong. To support Internet electronic commerce, modern web browsers are shipped pre-configured with a list of "certification authorities" and their public keys, which can be used to verify other certificates that have been signed by those authorities. This process is potentially recursive, since certification authorities can delegate their authority to others who can sign certificates and so on. In this manner, to confirm identity, one only needs to trust a few "top level" authorities.

This sort of hierarchical delegation and control of authority appeals to governments and bureaucracies. However, there is nothing to prevent anyone from installing their own certifying authority, say in the IT department of a hospital, and setting up the clients and servers of the TLS transactions to recognize certificates signed by their own local authority. Alternatively, one can easily outsource such a facility. One can even easily install new certificates in off-the-shelf web browsers.

Notice that this process can reliably establish the identity of both the server (or in DICOM terminology, the association acceptor, usually the SCP) and the client (association initiator, usually the SCU), with minimal infrastructure in place for distribution of certificates.

Whether that "identity" actually belongs to a person, or is assigned to a device, is up to the application. Within a hospital, if the local network is not trusted and secure communication between devices is required, TLS can be used between modalities, workstations and archives by using certificates assigned to devices. In a more distributed setting, certificates can be assigned to individuals for the purpose of controlling their access. Access can be restricted by refusing to establish TLS connections with individuals who are either unrecognized, or recognized but not authorized.

---

8. *Which is why the private key must be guarded ... if it is let out, then the "possessor" is no longer necessarily the "owner."*

In other words, TLS can be used to participate in a more generic access control framework, by potentially providing reliable, unique, identification of individuals.

The question then arises as to how individuals get these certificates in the first place. There are many potential approaches, including:

- ▶ “statically” assigning a certificate and storing it on something the individual possesses, such as on a particular computer, or on a floppy disk
- ▶ “dynamically” assigning a certificate based on some other form of authentication, such as something the individual is (a physical characteristic), something they know or something they possess.

In the latter case, one may use:

- ▶ some form of biometry, such as a fingerprint, face or retinal scanner
- ▶ a secret that they know, such as a conventional user-password combination, a PIN number, or a challenge-response protocol
- ▶ a device that they possess, such as a smart card, a key card, or a one-time password generator synchronized with the server

In other words, TLS can be used within any other infrastructure that provides for authentication of an individual’s identity, with the resulting certificate used to prove identity during communication. Such communication may then be restricted on the basis of that identity.

As an example, consider the case where a renal physician is called to consult on a post-operative patient whose kidneys have failed. Since the patient has been admitted under the general surgical service, normally only surgical staff have the right to access such a patient’s record. However, the act of initiating the consult has triggered the system to allow short-term, restricted access to the renal team to view (but not alter) the patient’s record, results and images, as well as the right to order diagnostic investigations and limited therapeutic interventions.

What is being described is a relatively complex and site-specific access control system. It is well beyond the scope of any existing health communications standard such as DICOM, but is well within the bounds of feasibility to construct as a proprietary application. What happens when the renal physician goes to call up the result and images of a renal ultrasound performed on the patient, stored as structured reports and images in a DICOM archive?

The renal physician logs on to the access control system by using their user name and password. Having satisfied itself as to the individual’s identity, the system issues a digital certificate in the individual’s name, signed by the access control system. The system “installs” the certificate in some shared context which is accessible to the image and reporting viewing application.<sup>9</sup> When the physician goes to query the DICOM server for the patient’s report and images, a TLS session is initiated. The viewing application first confirms that it really is talking to the server it thinks it is, by authenticat-

---

9. Such as a “cookie” or a CCOW context.

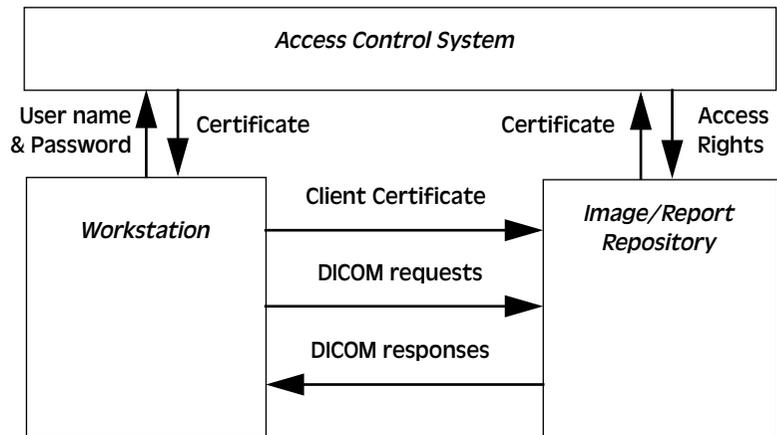


FIGURE 50. TLS, Authentication and Access Control

ing the server’s certificate. It then sends to the server the certificate issued by the access control system for the renal physician. The server confirms that the certificate sent by the client is a valid certificate (i.e. one signed by the access control system). The server then proceeds to determine what the identified individual’s rights are.

At this point the access control system confirms the physician is allowed to establish a connection, and allows the TLS session to proceed, after which a normal DICOM association is established “on top of” the secure TLS connection.

The physician now performs a query, looking for the patient in question. When the server (acting as a DICOM Query SCP) receives the query, it again consults the access control system which confirms that this individual is indeed permitted to view the reports and images of this particular patient. Subsequent retrieval and storage requests are similarly authorized and performed, and the physician goes on about their business.

The physician having carelessly forgotten to log out from the workstation, the device itself automatically does so (discarding the physician’s identity and certificate in the process), and awaits the next log in. In the meantime, the server has recorded in the audit trail all accesses to the patient’s data that were performed in the name of the renal physician.

Notice the points of contact between what one might regard as a “normal” DICOM viewing station, and the proprietary access control system:

- ▶ logging on provides information for the access control system to authenticate the user
- ▶ the access control system generates, signs and installs a certificate
- ▶ the viewing application uses the certificate to establish a DICOM session over TLS
- ▶ all query, retrieval, and storage transactions are performed using DICOM in the normal manner

The archive server, responding to the DICOM query, retrieval, and storage transactions, needs to consult the access control system only to determine the access rights of the individual whose identity is contained in the certificate passed in the TLS connection. It may do this when the connection is first established, or recheck the rights every time a service request is made on the open association. Once having established the access rights, it can proceed to allow or disallow associations, particular DICOM services, access to particular patients or types of information, as well as to log all accesses in its audit trail. Figure 50 illustrates the configuration described.

## **TLS, Confidentiality and Integrity**

Authenticating the identity of the communicating parties is only one aspect of the TLS protocol. The connection can also be made such that communications on the connection are protected from eavesdropping. This is done by encrypting all the transmitted information.

The details of how this is done are not really relevant here. Sufficeth to say, “session” keys are created and exchanged that allow for fast, efficient and secure symmetric encryption. The session keys are protected during their initial exchange by using the public keys contained within the certificates of the communicating parties.<sup>10</sup>

Not only is the encrypted information kept confidential, but any attempt to alter it in transit will be detected. A “message authentication code” (MAC) is created, which is a compact representation of the message combined with a shared, session-specific secret. At the receiving end, the MAC is recreated from the transmitted message and the shared secret, then compared with the transmitted MAC. If they do not match, the message has been altered in transit.

## **Security Features Enabled by DICOM and TLS**

Earlier in the chapter some of the important security requirements were listed. How far does the combination DICOM and TLS go in satisfying these requirements?

- ▶ authenticating individual identity: TLS allows communication of the identity of the principals<sup>11</sup> on both ends, given previously issued digital certificates
- ▶ access control: TLS can convey who (and/or where) a principal is; it is up to the server to control what access they are then granted
- ▶ audit trail: TLS can convey who the principal is; it is up to the server to log who made the access, to what it was, and when it was made
- ▶ confidentiality: TLS cryptographically protects information in transit
- ▶ digital signature: nothing to do with TLS; other mechanisms are necessary

---

*10. Public key cryptography is too slow to be used for large amounts of data, and hence faster symmetric ciphers are used instead. There are also ways to establish symmetric session keys other than to wrap them in public key encrypted containers.*

*11. “Principal” is used here to include either a device or a person.*

- ▶ integrity: TLS verifies the integrity of messages in transit only; it does nothing to verify that the object was valid *before* it was transmitted
- ▶ non-repudiation: TLS has no role to play in non-repudiation

Also, the use of TLS is purely confined to network transactions. TLS has no role in the security of objects recorded on interchange media.

## Digital Signatures

So far the issues of transmitting an object and controlling access to it have been addressed. A separate issue is confirming that an object really was created by a particular “author” and that it has not been altered. In other words, how can one ascertain the veracity of a document stored in a DICOM object?

In the past, this has not been an issue for DICOM. No-one seriously expects malicious individuals to run around editing images in Adobe Photoshop® and reinserting them into the system to have them masquerade as genuine patient data. There is a certain level of trust in a physically self-contained hospital image network and archiving system. The introduction of structured reporting has the potential to change all that. Traditional paper reports have a certain legal status as a document. They are part of the patient’s medical record. For many years, physicians have entered, corrected or verified their reports using electronic systems. Their “electronic signature”<sup>12</sup> has been accepted as being equivalent to a genuine hand-written signature on a paper document, certainly for the purpose of reimbursement. In the majority of cases, this “electronic signature” is implemented as a flag stored in the database, the signer’s identity having been validated to the extent of confirming their user name and password. It has been unusual up until now to implement this mechanism with a cryptographic form of digital signature, since the reports have resided within a single self-contained system.

If documents are to be persistent and distributed outside of a single system, then a better form of “signature” is needed. It is important to be able to verify that a document transmitted across the network or stored on interchange media was:

- ▶ created by the person (or device) who claims to be its author
- ▶ has not been altered
- ▶ was created at the date and time that is claimed

The use of public key cryptography, as described in the previous section, provides a convenient mechanism to achieve this.

A very compact equivalent of the contents of the document can be created by passing the document through a “one-way hash function”<sup>13</sup> to create what is referred to as a “message digest.” This message digest can then be encrypted using the author’s pri-

---

12. Interestingly, people new to the field often expect an “electronic signature” to be a scanned in bitmap or similar representation of a hand-written signature. This is very rarely the case, even for documents that are subsequently printed or faxed. The use of “signature” is used in a more general sense here.

vate key to create a digital signature. Anyone receiving the message can decrypt the digital signature using the author's public key (which everyone is allowed to know), and compare the decrypted message digest with the message digest that they produced themselves from the message they actually received. If the two match, then the message has not been altered since it was "signed," and only someone possessing the private key (presumably the author) could have signed it.

To verify the signature, the receiver must have the public key of the signer. This can in theory be made available from some central or distributed system, but there is an easier way. The message can be accompanied by a "digital certificate" that contains the public key of the signer linked to their identifying information.

## Digital Certificates

How does a receiver know that a certificate really does belong to the person identified, and isn't someone else masquerading as that person?

The answer is that a certificate itself may be signed, this time by someone well known to the recipient, such as the hospital or a publicly known "certifying authority." This is just the same as was described in the section on TLS. Indeed whenever public key cryptography is used as part of a security solution, be it for digital signatures or network security or any other application, an infrastructure for the distribution of digital certificates is required. This is referred to as a Public Key Infrastructure (PKI), the specifics of which are sometimes hotly debated.<sup>14</sup> In common use however, are:

- 
13. A one-way hash function takes a whole lot of data and "hashes" it into a short number, usually 128 or 160 bits. It is almost impossible to create a similar message that will result in the same hash value, especially by altering only a few bits of the source data. For example, changing a name or a conclusion or an observation in an SR document would result in a completely different hash value from that which was computed from the original document.
  14. Some people are in favor of hierarchies, such that trusted people at the "top" confirm identities and delegate authority to those below them, recursively. Others are not so trusting of hierarchies and fear big-brother style regulation. They prefer a "web of trust" approach, such that if an identity is confirmed by enough others who are trusted (e.g. friends and colleagues), then they probably trust it too. This is the approach used by the Pretty Good Privacy (PGP) software, but it is probably not practical for health-care applications.

*Interestingly enough, in the US, various organizations are proposing to provide certificates and authentication services for physicians. The AMA has teamed up with Intel, for example, and proposes to provide free "digital credentials" for all physicians (including non-members), though "other parties will pay licensing or transaction fees to Intel" according to the AMNews, June 26, 2000. See "[http://www.ama-assn.org/sci-pubs/amnews/pick\\_00/tesb0626.htm](http://www.ama-assn.org/sci-pubs/amnews/pick_00/tesb0626.htm)". Whether the federal government will consider mandating the use of the Intel-AMA approach, for reimbursement or other purposes, remains to be seen.*

*It turns out the AMA keeps a record called the "Physician Masterfile Database," which "includes 850,000 uniquely identified records of AMA members, non-members, and all active and inactive physicians in the US and territories." According to testimony to the National Committee on Vital and Health Statistics (NCVHS) Subcommittee on Privacy and Confidentiality, this file has been made available to Intel. Apparently, physicians are not as deserving of "privacy and confidentiality" as are patients. See "<http://www.ama-assn.org/med-sci/cpt/finalncvhs.htm>".*

- ▶ a standard form in which to encode the certificates, X.509<sup>15</sup>
- ▶ certificates signed by “certifying authorities,” whose own certificates are trusted and installed in off-the-shelf software, such as web browsers

The deployment of systems based on digital certificates is non-trivial. There are many issues to be faced such as:

- ▶ authenticating the identity of an individual before issuing a certificate
- ▶ the length of time a certificate should be valid before it expires
- ▶ how to “revoke” a certificate that is no longer valid
- ▶ how to distribute and install certificates

## Digital Signatures in DICOM

How a document, or for that matter, any stream of bits, can be digitally signed has been described. How does this apply to DICOM composite objects, particularly structured reports?

Several approaches are possible:

- ▶ sign an entire DICOM document as one entity
- ▶ sign selected parts of the data set

In either case, it is conceivable that one might want to attach multiple signatures to the same document.

The “sign the whole thing” approach was considered because it would be so easy, but was rejected in favor of signing selected parts of the document. The reasons for this included requirements that:

- ▶ different parties might need to take responsibility for different components within the document
- ▶ some of the “management” information might need to be altered within a system, without invalidating signatures that applied to the primary content

The details that follow are derived from the most recent draft<sup>16</sup> of the proposal to add digital signatures to DICOM. Obviously, implementers should use what is described in any final standard, rather than any information provided here in the interim.

---

15. X.509 is an ITU standard. Internally, X.509 certificates are encoded using the Distinguished Encoding Rules (DER) of Abstract Syntax Notation 1 (ASN.1) so one also needs X.680-683 and X.690. Fortunately, sufficient material is documented in books on the subject, so that those sufficiently masochistic to actually want to implement X.509 by hand probably don't need the ITU standards. See Stephen Thomas's book on “SSL an TLS Essentials” already mentioned, or “Feghhi J et al. Digital Certificates. Addison-Wesley. ISBN 0-201-30980-7.”

16. Specifically, Supplement 41 version 0.7, dated 30 May 2000.



DICOM objects are encoded in data sets that consist of a sorted list of data elements that correspond to the attributes of the object. One or more digital signatures may be added to the data set, such that each signature contains, amongst other things:

- ▶ the digital certificate of the signer
- ▶ the list of data elements included in the signature
- ▶ the signature itself

This mechanism allows for multiple parties to sign a document, each of whom can sign a different list of data elements. Various different algorithms can be used in hashing the values of the data elements to produce the message digest<sup>17</sup> that is signed. Different algorithms can be used to produce the signature from the hashed value.

To improve conformance, “security profiles” are specified which define:

- ▶ what list of data elements are (at a minimum) to be included for particular purposes, such as “creation” or “authorization” of the document
- ▶ what algorithms are used for the MAC calculation and the signature, such as RIPEMD-160<sup>18</sup> and RSA<sup>19</sup>

For example, the proposed “creator RSA signature profile” specifies that the following data elements must be signed:

- ▶ the SOP Class and Instance UIDs
- ▶ the SOP Creation Date and Time, if present
- ▶ the Study and Series Instance UIDs
- ▶ any attributes of the General Equipment module that are present
- ▶ any overlay data present
- ▶ any image data present

For the purposes of structured reporting, to this list should probably be added various attributes of the SR Document General Module such as those for verifying the object, as well as the content of the document itself. Specifically, the top level content item attributes that are not contained in a sequence should be included, as well as the content sequence that contains the children of the root node.

The digital signature is applied to the MAC which is calculated from various parts of each encoded data element. Specifically:

---

17. Or “Message Authentication Code” (MAC).

18. RIPEMD-160 was chosen over other commonly used hash algorithms like MD-5 and SHA, since MD-5 apparently has some weaknesses and there is concern over the US government involvement in the definition of SHA. The decision as to which algorithm to use may be revisited.

19. Though the RSA algorithm was formerly patented in the US (only), it is widely used throughout the world. The US patent expired on September 20, 2000. The algorithm is now free for use anywhere. Indeed, RSA Security Inc. placed the algorithm in the public domain two weeks before the patent expired.

- ▶ the data elements to be included are fed into the hash function in the order in which they are encoded in the data set (that is sorted by ascending group and element number)
- ▶ the data elements are hashed as if they were encoded in an uncompressed little endian transfer syntax<sup>20</sup>
- ▶ only the group and element tag and the value of the element are fed to the hash function, not the explicit value representation nor the value length
- ▶ sequences can only be signed in their entirety, and when doing so the item tags as well as the contained data element tags and values (but not any explicit lengths or delimiters) are included

Notice that since a sequence item is itself an entire DICOM data set, digital signatures can be included within sequence items. For structured reports, this means that it is possible to apply digital signatures to selected parts of the content tree, by selectively signing content sequence items (and all their included content). Whether or not this will be feasible or useful remains to be seen. In practice, it is likely that only the top level data set will ever be signed, and any sequences will be included in their entirety.

Since bulk data (such as waveforms data and pixel data) may be included in the MAC, the tags that compose the digital signature are organized so that the list of what data elements are to be signed precedes the bulk data, and the signature itself follows them.<sup>21</sup>

The certificate of the signer is included in the data set, encoded in the conventional X.509 format. There is also provision to include an optional time stamp to indicate when the signature was applied. However, the reliability of any time stamp is only as good as the creator of the data claims it to be, in the absence of a trusted third party time stamping system. The DICOM field allows for a reliable time stamp.<sup>22</sup>

## Validating Signatures and Bit Preservation

The process of validating the digital signatures in a DICOM object includes:

- ▶ verifying the authenticity of the signer's certificate
- ▶ extracting the list of data elements signed
- ▶ making a local MAC, by feeding those elements which were signed into the hash algorithm, just as the creator of the signature did

---

20. Lossy compressed pixel data cannot be signed, since there is no way to verify the signature subsequently.

21. This is one of the few cases in which there will be data elements that follow the pixel data element (7FE0,0010).

22. The most recent version of Supplement 41 specifies the format described in "Adams C et al. Internet Draft: Internet X.509 Public Key Infrastructure. Time Stamp Protocol (TSP). June 2000." How a time stamp is obtained is not defined by the standard. As it turns out, the basic patents on time stamping seem to be concentrated in the hands of one company that also happens to provide a time stamping service on a fee per transaction basis.

- ▶ comparing the computed MAC with the decrypted MAC from the signature

If the values of each MAC are the same then:

- ▶ whoever created the object and signed it did indeed possess the private key corresponding to the certificate, and should be the person identified in the certificate
- ▶ the document has not been altered since it was signed (or at least those parts of the document that were included in the MAC have not been altered)

Notice that for this to work, every single bit of every single data element incorporated in the signature must be fed to the hash algorithm in exactly the same order at both ends. Though data elements not included in the signature may have been altered, added or removed, none of those that have been signed can have been altered in any way.

This also means that the values of data elements may not have been transformed in any way, such as by adding or removing otherwise insignificant padding, changing lower case names to upper case, etc. In other words, the object must have been transferred and stored in a “bit preserving manner.”

Many existing DICOM implementations are not bit preserving, and routinely “clean up” inbound attributes, normalize them for storage in data bases, and reconstruct the DICOM objects from an internal representation when they are requested. Indeed many systems don’t even preserve all the data elements they receive<sup>23</sup>.

Systems that are not bit preserving cannot preserve digital signatures. The best they can do is verify the signature when an object is received, store the fact that the original signatures were valid, and then sign the object again themselves when recreating or exporting the object. This is a so-called “proxy signature.” In such cases, one has to trust the system to have faithfully checked the inbound signatures.

Because it is important to know whether or not a system is bit preserving in the context of digital signatures, additional flags have been added to the DICOM association negotiation mechanism to signal this.

## **DICOM Media Security**

The digital signature mechanism just described can be used for DICOM objects stored on interchange media, just as easily as it can for objects in a repository or transferred over the network.

However, the confidentiality and peer authentication features of TLS cannot be used with media, since a live connection is required for the TLS protocol.

Objects stored on media are vulnerable to interception or alteration if the media falls into the wrong hands. This has not been considered much of a practical problem,

---

*23. There are different “levels” of DICOM storage service class provider defined, that can be determined during negotiation at association establishment time. In the case of structured reports, all storage service class SCPs are required to be “level 2,” which means that all standard required and optional attributes must be stored.*

since interchange media is no different from printed records or images on film. The standard of practice for protecting physical records is to:

- ▶ keep them physical secure while under the control of the health-care provider
- ▶ deliver them into the hands of the patient who then assumes responsibility for them
- ▶ transfer them between providers using a trusted courier mechanism

There is no special reason that the requirements for digital physical media should be any different, were it not for the mere fact that the technology exists to do a better job of it. People just seem to have a heightened awareness of the vulnerability of digital information, coupled with the knowledge that cryptographic solutions are available.

The application of public key cryptography to digital physical media allows for the possibilities of:

- ▶ protecting the confidentiality of the information
- ▶ restricting the recipients by whom it is readable

As has been mentioned, it is impractical to use public key algorithms to encrypt bulk data, so symmetric algorithms are used, a unique key being generated each time this is done (a “session key”). Given that this key needs to be transmitted with the media, and hence itself encrypted in a public key protected “envelope,” the recipient must possess the corresponding private key to decrypt the session key and thence the data.

One nice side effect of this approach is that the same session key can be wrapped in multiple envelopes for multiple recipients. Indeed some information on the media can be made available to some recipients but not others.

The remaining question is then whether or not to protect the information:

- ▶ at the physical level, such that every sector on the media is encrypted by the hardware or the device driver
- ▶ at the file system level, such that every file and the file system directory is protected by the operating system
- ▶ at the individual DICOM file (and DICOMDIR) level, by a secure application
- ▶ at the data element level, within each DICOM file

If every removable disk drive or computer or operating system came with such encryption facilities, and they were interoperable across different platforms, then DICOM would simply pick one and be done with it. Unfortunately this is not yet the case,<sup>24</sup> so DICOM is considering adding its own solution at the file and data element

---

24. Though recently a proposal to use Secure UDF has been received. It remains to be seen whether or not operating system and optical media vendors support Secure UDF for consumer applications.



levels. Rather than reinvent the wheel, options popular on the Internet such as Cryptographic Message Syntax (CMS)<sup>25</sup> are being evaluated.

It is fair to say at this point that there is no current DICOM solution for media security, but the matter is being pursued.

## **Legal and Regulatory Issues**

No chapter on security would be complete without at least mentioning regulatory and legal issues. Since these are in a constant state of flux, the material here will date quickly, so this section will be brief.

Regulatory issues basically fit into the following categories:

- ▶ rules that restrict or forbid domestic use of encryption
- ▶ rules that restrict or forbid export of encryption technology
- ▶ rules that mandate law enforcement access to encrypted material
- ▶ rules that require protection of a patient's information in terms of access control and confidentiality
- ▶ rules that determine the evidentiary value of digitally signed documents
- ▶ rules that determine reimbursement for procedures resulting in digital rather than printed or filmed information

This is not an exhaustive list. It may convey a sense of the morass of bureaucracy that a vendor has to wade through in order to implement products that incorporate security features.

Until recently, in the US the situation was that one couldn't export any cryptographic solution worth using, patients had no particular right to privacy or protection, and digitally signed documents were not recognized as having the same evidentiary standing as paper documents. This situation is changing as:

- ▶ export controls for "strong" cryptography are gradually being relaxed to allow US companies to be competitive<sup>26</sup>
- ▶ the HIPAA final rules on standards for privacy as well as security loom nearer<sup>27</sup>
- ▶ digital signatures are recognized in law, thanks to recent legislation<sup>28</sup>

Interestingly, the pendulum in the US has swung from one extreme to the other. Formerly, there was a total lack of interest in security for health-care applications. Now there is near hysteria, to the point that one reads estimates that more money will be spent on "HIPAA compliance" than on the Y2K problem. It does seem high time that some attention was paid to the woefully inadequate existing state of security in health care. Hopefully, any provisions that are enacted will not be so restrictive as to interfere with the safe practice of medicine. There is a significant risk, however, that costs

---

25. Described in RFC 2630.

26. See for example 15 CFR Parts 734, 740, 742, 770, 772, and 774 which amend the Export Administration Regulations (EAR).

will increase to levels that are ridiculous relative to the real (as opposed to perceived) level of threat.

In Europe, attention has been focused on the rights of individuals and patients to privacy for some time, as a consequence of a directive protecting personal data.<sup>29</sup> The digital signature law in Germany was amongst the first to recognize the importance of digital signatures. There is now also European legislation providing for a framework for the use of digital signatures.<sup>30</sup> Interestingly enough, not all countries in Europe are as willing as others to allow their citizens access to cryptographic solutions for privacy, leading to some interest difficulties for vendors. Though it has little impact on health-care applications, the governments of some European nations (such as Britain) expect to retain the ability to monitor their subjects' communications en masse for law enforcement and national security purposes, much as many people believe the US government also does.

In Japan, the Ministry of Health and Welfare has revised regulations on the security of "electronic storage" both for off-line media and on-line exchanges. In the past, highly specific technologies had been mandated, but more recently, regulations that specify requirements rather than solutions have been issued.<sup>31</sup>

## Summary

Some of the key points mentioned in this chapter include:

- ▶ security of reports, as well as images, is likely to become of greater importance, particularly as exchange of health-care information over the Internet becomes more widespread
- ▶ DICOM provides support for data integrity, confidentiality and mutual peer authentication by using the Internet standard TLS (SSL) protocol

---

27. *Health Insurance Portability and Accountability Act (HIPAA) of 1996. HR 3103. One may well ask what insurance has to do with security. It goes something like this: people change jobs; employers provide health insurance; next employer's insurer must pick up new employee; patient's information needs to be exchanged between insurers; desire to promote electronic transfer of such; require secure transfer of such; any "partners" on either end of such transfer might compromise privacy (i.e. everyone involved in health-care provision); therefore HIPAA purportedly gives mandate to regulate privacy of practically all health-care transactions, insurance related or not. Go figure.*

*See also the HIPAA proposed rule, 45 CFR Parts 160-164 Standards for Privacy of Individually Identifiable Health Information and the HIPAA proposed rule 45 CFR Part 142 Security and Electronic Signature Standards.*

28. S 761, the Millennium Digital Commerce Act.

29. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 "on the protection of individuals with regard to the processing of personal data and on the free movement of such data."

30. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 "on a community framework for electronic signatures."

31. Director-General, Health Policy Bureau. "The Electronic Storage of Clinical Records." HPB No. 517. April 22, 1999.

- ▶ TLS can be used to facilitate control of access to DICOM services by using digital certificates that reliably identify unique individuals
- ▶ an extension to DICOM is being developed to incorporate digital signatures at the data element (attribute) level
- ▶ DICOM is investigating ways of securing information on media at the file system, file and data element level



## ***Other Standards: HL7, MIME, XML, COAS***

---

DICOM is not the only standard for communication of health-care related messages. The domain of DICOM is largely restricted to images and associated information, as the acronym suggests.<sup>1</sup> In the imaging domain, the standard has had unparalleled, international, success. Outside the world of imaging there are other standards. The most prominent of these (that is health-care specific) is the Health Level 7 (HL7) standard. HL7 addresses the need to standardize the format and meaning of messages exchanged between information systems, such as admission, discharge and transfer (ADT), order entry, scheduling, and so on. Though many information systems do not “natively” communicate messages in HL7 format, most provide HL7 interfaces which allow such messages to be communicated to and from the external world.

Now that DICOM has seriously entered the field of reporting,<sup>2</sup> a dilemma arises. Reports related to images may well be created in the imaging department, and are likely to contain embedded references to those images. One must, however, face the reality that reports are normally managed and distributed outside the imaging department by so-called “information systems” (IS). What is a poor IS to do when faced with a DICOM encoded structured report?

There are a number of alternatives:

- ▶ extend the use of DICOM encoding and protocols beyond the confines of the imaging department to the rest of the enterprise
- ▶ use a web based distribution mechanism which serves images and reports from within the imaging department, but is accessible to conventional clients (browsers) throughout the rest of the enterprise

---

1. *Digital Imaging and COmmunications in Medicine.*

2. *The services for reporting that existed in DICOM before the advent of structured reporting were limited and not widely adopted.*

- ▶ embed DICOM reports and images within messages that can be transmitted to the IS (such as HL7 messages), which can distribute them to upgraded IS terminals and applications which are capable of rendering the embedded (DICOM) content
- ▶ transcode the DICOM reports (and conceivably images) into a form that is directly usable by the IS, which can then distribute them to its existing terminals and applications

The last two approaches, embedding and transcoding, will be covered in detail in this chapter. Mechanisms for distribution of reports by e-mail and the web, as well as using other standards like OMG COAS<sup>3</sup> will also be mentioned.<sup>4</sup> In particular, the use of XML will be examined, both as a means of encoding structured reports in general and as the “implementable technology” chosen for the HL7 Clinical Document Architecture.

## HL7 101

This is not the place for an extensive tutorial on HL7. However a few basic concepts will be described for those with no previous experience with the standard.

HL7 comes in two basic flavors, version 2.x<sup>5</sup> and version 3. Almost all existing implementations are of version 2.x. This is not surprising, since version 3 isn’t completely finished yet. Accordingly, the discussion in this chapter will focus solely on 2.x, with references to more recent features only when they may be conveyed in 2.x messages.<sup>6</sup>

Unlike DICOM, which specifies a standard for multiple levels of the ISO OSI model and has strict conformance requirements, HL7 takes a completely different approach. It addresses the “7th” level of the OSI model, the application layer, hence the name “Health Level 7.” The format and meaning of messages are defined in detail, as are the circumstances under which they will be sent (trigger events). However, very few fields are mandatory, there are no conformance requirements, and there is no exchange protocol defined. Two implementations that wish to communicate must reach an a priori agreement on how messages will be exchanged, what real-world activities will correspond to the trigger events, and what fields the messages will contain. For example, two devices might agree to communicate over a serial port at a certain bit rate, with certain special characters indicating the beginning and end of a message. Two other devices might communicate over a network using TCP/IP, with the so-called Minimal Lower Level Protocol (MLLP).<sup>7</sup>

---

3. *Clinical Observations Access Service.*

4. *Unfortunately, it isn’t possible to describe every standard or proposed standard that has merit. In addition to those mentioned in this chapter, the reader may care to study the CEN SCP-ECG standard, which provides a framework for structuring reports about ECGs. CEN TC251 is also working on standards for the electronic health-care record and multimedia reports.*

5. *At the time this was written, the most recent version published was 2.4.*

6. *This includes the Clinical Document Architecture (CDA).*

HL7 messages are composed of “segments” separated by carriage returns. The first segment is always the MSH or message header segment, which, amongst other things, assigns an ID to the message, specifies the separator (delimiter) characters used<sup>8</sup>, and specifies the “trigger event” that stimulated the message to be sent. Subsequent segments carry the payload of a message. Many segments are common to multiple different types of message. In a sense, HL7 message definitions are analogous to DICOM information object definitions, and HL7 segments are analogous to modules. Just as DICOM modules consist of attributes that are encoded as data elements with various value representations, HL7 segments are composed of “fields” that have a “data type” associated with them. There are no explicitly conveyed tags to identify a field in a segment, and no explicit data type conveyed. The meaning of a field is conveyed by its position in a segment alone.

There is a vast range of data types specified in HL7, some of which correspond closely to the value representations used in DICOM, though of course there are no binary data types. HL7 messages are strictly ASCII only. Later, in the section on transcoding, the correspondence between HL7 data types and DICOM value representations will be revisited, particularly with respect to names of people, object identifiers and coded entries.

A simple example of an ADT admit/visit notification (A01) message may serve to illustrate some of these characteristics:<sup>9</sup>

```
MSH|^~\&|ADT|ST.ELSE|XRAY|ST.ELSE|200008091837|
|ADT^A01|MSG0001|P|2.3.1|\r
EVN|A01|200008091820|\r
PID||MRN1234|||SMITH^JOHN||19501009|M||C
|220 WHITE PLAINS RD^^TARRYTOWN^NY^10591
|(914)332-4800|\r
NK1||SMITH^MARY|WI^WIFE|||NK^NEXT OF KIN\r
PV1||I|CATHLAB|||919191^CLUNIE^D.||||RAD|||ADM|AO|\r
OBX||ST|1010.1^BODY WEIGHT||62|kg||||F\r
DG1||19|BIOPSY||00|\r
```

In this example, notice that:

- ▶ the message type and trigger event (ADT^A01) are conveyed in the MSH segment, and the trigger event is repeated in the EVN segment

7. The MLLP is described in the Implementation Guide, and is not, strictly speaking, part of the HL7 standard itself. Nor is it the only protocol for exchanging HL7 messages over TCP/IP, nor is it a complete specification in that regard.
8. Most people are used to seeing HL7 messages with fields in segments delimited by ‘|’, components in fields separated by ‘^’, repeated fields separated by ‘~’ and a ‘\’ used as the escape character. However, this convention may be changed in any message by specifying different delimiters at the start of the MSH segment.
9. In this example, message segments are separated by ‘\r’ and have been artificially continued on to separate lines to fit the page; continuation is indicated by indentation.

- ▶ segments are identified by three character letters, such as PID for patient identifier, NK1 for next-of-kin, PV1 for patient visit, OBX for observation and DG1 for diagnosis
- ▶ various data types are in use, including those to describe names like “SMITH^JOHN”, coded entries like “1010.1^BODY WEIGHT”, and date-times like “200008091820”

The OBX segment is of particular interest, as will be further discussed in the section on encapsulation.

## Encapsulation in HL7 Messages

Traditional reports in an information system are conveyed in OBX segments of MDM or ORU messages. OBX segments are used to convey all sorts of different kinds of observations, including atomic observations of individual properties, as in the example of the previous section:

```
OBX||ST|1010.1^BODY WEIGHT||62|kg||||F\r
```

If a report consists of plain ASCII text it can be transmitted in a similar manner:

```
OBX||TX|^Chest X-Ray Report||Lung fields clear.\r
```

Notice how the second field of the OBX segment<sup>10</sup> specifies the value type of what is to follow.

If one wanted to encode a waveform in HL7, rather than as a DICOM object, it would be contained in OBX segments, encoded according to an extensive set of requirements, and using waveform-specific value types such as numeric array (NA), multiplexed array (MA) and channel definition (CD).

To encapsulate arbitrary data, one can use the ED (“encapsulated data”) value type in an OBX segment. Encapsulated data may be sent as ASCII, hexadecimal bytes (octets) or encoded as Base64.<sup>11</sup> This is similar in principle to mechanisms like “uuencoding” that have been used in e-mail and Usenet newsgroups for many years, but is more compact (or rather, less expanded).<sup>12</sup> Encapsulating a DICOM object, such as an image, in an ED data type in an OBX segment, using Base 64 encoding might look something like this:

```
OBX||ED|209049^Baseline image^99PMP|
|^Image^DICOM^Base64^
```

---

10. “OBX.2 field 00570 Value Type” in the nomenclature of the HL7 standard.

11. In Base64 encoding, three binary bytes are encoded as four ASCII characters, using upper and lower case alphabetic characters, digits ‘0’ through ‘9’, ‘+’, ‘/’ and “=”.

12. One may well asks why transmitting binary information has been such an issue, and transcoding to ASCII necessary. The issue is that ASCII fits in 7 bits, and over serial lines the 8th bit was often used for parity, and not available for data transmission. This hardware restriction has largely disappeared, but ostensibly 7 bit limits persist in some e-mail gateways, leaving an irritating legacy to work around. Interestingly, the popularity of XML has revived obsession with plain text encoding of binary data.

```
byEAALcAAABbAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
... rest of Base64 encoded DICOM object ...
|\r
```

Notice that when encapsulating data in an ED data type in an OBX segment:

- ▶ OBX.2 field 00570 Value Type is set to ED
- ▶ there is still a coded entry (CE) label for the observation in OBX.3
- ▶ the type of content and the actual content are all sent in a single field, OBX.5 (field 00573 Observation Value) which is separated into multiple “components”

The individual components of the 5th field are specified as follows:

- ▶ the 2nd component is set to the type of data (e.g. “Image”)
- ▶ the 3rd component is set to the sub-type of data (e.g. “DICOM”)<sup>13</sup>
- ▶ the 4th component is set to “Base64”
- ▶ the 5th component contains the actual Base64 encoded data<sup>14</sup>

## MIME Encapsulation

The use of the ED data type in the previous section is sufficient to encapsulate a single binary object. A more general mechanism that allows for the encapsulation of multiple objects in the same observation is also available. This approach makes use of the e-mail and web convention for packaging and encoding objects, the Multipurpose Internet Mail Extensions (MIME).<sup>15</sup>

The MIME specification includes a “content type” which further describes what is contained. There are standard content types such as “text/plain”, “image/gif” and so on. MIME content types are widely used on the internet to signal what content is contained in e-mail attachments, as well what kind of content is present in an http message.<sup>16</sup> MIME messages that are ASCII text may be used in HL7 OBX segments with an ED data type by specifying a type of “multipart”, an encoding of “A” for ASCII and sending the whole MIME message as the data value.

DICOM has recently requested that the MIME content type “application/dicom” be registered to describe the presence of any DICOM SOP class in a message.<sup>17</sup> The pri-



13. In HL7, the sub-type of DICOM specifies a PS 3.10 file with a File Meta Information header.

14. Any HL7 separator characters that occur in the Base64 encoded data need to be escaped. No new lines are used; the encoded characters are sent as a continuous stream.

15. Described in RFC 2046.

16. They are even increasingly used in operating systems and window managers to associate types of files with appropriate applications.

17. More specifically, to describe any DICOM composite SOP class instance with a PS 3.10 File Meta Information header.

Until the “application/dicom” content type is formally approved, the experimental “application/x-dicom” content type can be used.

mary intention is to allow DICOM objects to be sent as e-mail attachments. However, given the availability of an appropriate content type, any DICOM composite object, even an image or a waveform, can in principle be wrapped in a MIME message, and then encoded in an HL7 OBX segment by using the ED data type.

The MIME message not only specifies what is contained in a message part, but also how it is encoded (and, potentially, compressed). Binary objects like DICOM images, waveforms and structured reports can be encoded in MIME text messages by using Base64 encoding.<sup>18</sup> A MIME package can consist of multiple parts, which need not be of the same content type. Internally, parts of a MIME message are separated by a “boundary.” This is a recognition string that occurs on a line by itself between parts and after the last part. The string to use for the boundary is specified as an attribute of the “top” content type header line, and is generally something weird, long, unique and not likely to occur in the encoded content.

Encapsulating multiple DICOM objects in an ED data type in an OBX segment using a MIME multipart message with Base 64 encoding might look something like this:<sup>19</sup>

```
OBX||ED|209073^Report with images^99PMP|
|^multipart^DICOM^^A^
  MIME-Version: 1.0
  Content-Type: multipart/mixed; boundary="--xx"
  Content-Transfer-Encoding: Base64
  --xx
  Content-Type: application/x-dicom; boundary="--xx"
  byEAALcAAABbAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  ... rest of Base64 encoded DICOM SR object ...
  --xx
  Content-Type: application/x-dicom; boundary="--xx"
  byEAALcAAABbAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  ... rest of Base64 encoded DICOM image object ...
  --xx
|\r
```

Notice that when encapsulating data in an ED data type with MIME:

- ▶ OBX.2 field 00570 Value Type is set to ED
- ▶ there is still a coded entry (CE) label for the observation in OBX.3
- ▶ the type of content and the actual content are all sent in a single field, OBX.5 (field 00573 Observation Value) which is separated into multiple “components”

The individual components of OBX.5 are specified as follows:

18. Indeed it is in the MIME RFC that Base64 encoding is defined.

19. This example is purely theoretical, and has not been tested by the author. It is not known whether or not there are any systems implemented that accept HL7 messages in this form and could make use of the encapsulated DICOM objects. These examples were extrapolated from information in Chapters 2 and 7 of HL7 V2.3.1 and the ballot text of HL7 CDA Level 1.

- ▶ the 2nd component, the type of data, is set to “multipart”<sup>20</sup>
- ▶ the 3rd component is set to the sub-type of data (e.g. “DICOM”)
- ▶ the 4th component is set to “A” for ASCII (not Base64)
- ▶ the 5th component contains the actual data (the MIME package)<sup>21</sup>

It may not make a lot of sense to encapsulate DICOM images in HL7, since they are generally orders of magnitude larger than normal HL7 messages. Encapsulating reports is much more realistic however. DICOM waveforms are generally not that large, though still larger than most HL7 messages. There is a specific HL7 mechanism for encoding waveforms, into which a DICOM waveform could be transcoded with little loss of fidelity.



**Not yet  
standardized.  
Final standard  
may differ  
substantially.**

## E-mail

To digress from the subject of HL7 for a moment, the MIME mechanism may also be used to attach DICOM objects to e-mail messages.<sup>22</sup> This usage is now (or soon will be) officially sanctioned by DICOM, though its inclusion in the standard was somewhat controversial. This is because the standard cannot specify any conformance requirements on which SOP classes and transfer syntaxes an e-mail recipient supports. Despite this, e-mail usage of DICOM objects is popular and widespread already. Accordingly, formally registering a MIME content type of “application/dicom” and describing the usage in the standard seems appropriate.

One or more DICOM objects such as images and structured reports can be attached to an e-mail message by using the multipart/mixed mechanism. The DICOM objects are required to be in PS 3.10 file format with a File Meta Information header. There are no constraints on the transfer syntax that may be used. Obviously uncompressed transfer syntaxes will be more likely to be readable by the recipient, but will be larger.

The proposed registration of the content type also refers to the notion of attaching a “file set” rather than a mere collection of individual files. A file set is defined in PS 3.10

---

20. The type “multipart” does not appear in the HL7 V2.3.1 specification. It is used, without further explanation, in the draft CDA document. Presumably, it will be defined in later versions of V2.x.

21. Any HL7 separator characters that occur in the MIME package need to be escaped, whether they occur in the MIME header or the Base64 encoded text. E.g. if the ‘|’ is the field separator and occurs anywhere in the message, it must be sent as ‘\|’, if ‘\’ is the escape character. The separator and escape character are specified in the MSH segment of every message. None of the default separators occur amongst the characters used by the Base64 encoding, but non-standard separators might.

New lines that are required to separate lines of the MIME package are encoded in HL7 by using escape sequences that are used for any text data type (even though the control/query chapter doesn’t specifically mention the ED and ST data types). Obviously a carriage return (‘\r’) character cannot be used since it separates HL7 segments. New lines in MIME messages are therefore sent as “\X0D0A\”.

22. As the name implies, Multipurpose Internet Mail Extensions were originally intended expressly for the purpose of encoding e-mail.

and includes a DICOMDIR file, which provides an index organized according to a patient/study/series/instance hierarchy.

An (abbreviated) example<sup>23</sup> of a MIME encoded e-mail message containing DICOM objects and a DICOMDIR follows:

```
From: "Dr Smith" <smith@provider1.com>
To: "Dr Johnson" <johnson@provider2.com>
Subject: DICOM File set MIME Example
Date: Tue, 29 Feb 2000 09:28:06 +0100
MIME-Version: 1.0
Content-Type: Multipart/mixed;
    boundary="-----_NextPart_000_0007_01BF8297.490E26C0"
```

This is a multi-part message in MIME format.

```
-----_NextPart_000_0007_01BF8297.490E26C0
Content-Type: text/plain;
    charset="Windows-1252"
Content-Transfer-Encoding: 7bit
```

Text of the message: this is a demo message of the DICOM MIME type, for a DICOM File set.

```
-----_NextPart_000_0007_01BF8297.490E26C0
Content-Type: Multipart/mixed;
    boundary="-----_NextPart_000_0007_01BF8297.490E26C1"
```

This is a multi-part message in MIME format corresponding to a "DICOM File set" MIME Entity.

```
-----_NextPart_000_0007_01BF8297.490E26C1
Content-Type: Application/dicom;
    id="i00023"; name="i00023.dcm"
Content-Transfer-Encoding: base64
```

```
byEAALcAAABbAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
...=
```

```
-----_NextPart_000_0007_01BF8297.490E26C1
Content-Type: Application/dicom;
    id="DICOMDIR"; name="DICOMDIR"
Content-Transfer-Encoding: base64
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
...=
```

---

23. This example is truncated from Annex B of Supplement 54, draft for letter ballot, dated 27 September 2000.

-----=\_NextPart\_000\_0007\_01BF8297.490E26C1--

-----=\_NextPart\_000\_0007\_01BF8297.490E26C0--

## HTTP, MIME and DICOM

It does not take a rocket scientist to deduce that since:

- ▶ there is a DICOM MIME content type (“application/dicom”)
- ▶ the HTTP protocol uses MIME encoding for the body of messages

then DICOM objects can readily be exchanged using the http protocol. This is not an officially sanctioned mechanism for exchange of DICOM objects. One cannot claim DICOM “conformance” if one uses http, since conformance claims are restricted to the DICOM protocol or interchange media application profiles. However, this approach has its uses, particularly when distributing information that is “part of” a web page and can be rendered using a web browser plug-in or applet that can handle the encapsulated DICOM content.

Indeed, for those web browsers that also act as e-mail clients, exactly the same mechanism is used to specify the behavior based on the MIME content type, whether the source of the content is an e-mail message or a web page.

## Transcoding and HL7

Encapsulating a DICOM encoded structured report and sending it to an IS is not going to be very useful unless the IS applications are “DICOM-aware.” In the majority of cases, they will not be able to make use of the object. A more practical strategy in many cases is to convert the DICOM object into something more palatable.

For many existing systems this means translating it into plain text. The approach is no different from rendering a structured report into plain text for any other reason, such as printing or display. Nested containers used as headings can be rendered as an “outline,” plain text can be rendered in place and text meanings can be inserted for coded entries. For example, the following DICOM SR:

```
<CONTAINER:(“Chest X-ray Report”)>
  <contains CONTAINER:(“Description of procedure”)>
    <contains TEXT:(“Description”)="PA, lateral views">
  <contains CONTAINER:(“Findings”)>
    <contains TEXT:(“Finding”)="Blunting of the left
      costo-phrenic angle, cardiomegaly
      and interstitial lines">
  <contains CONTAINER:(“Conclusions”)>
    <contains TEXT:(“Conclusion”)="Pulmonary oedema and
      pleural effusion">
  <contains CODE:“Diagnosis”=(428.1,I9C,
    “Left heart failure”)>
```

could be translated into a plain text message in an OBX segment something like:<sup>24</sup>

```
OBX||TX|^Chest X-Ray Report||
  Description of procedure.~
  - PA, lateral views were obtained~
  Findings.~
  - Blunting of the left costo-phrenic
    angle, cardiomegaly and interstitial lines~
  Conclusions.~
  - Pulmonary oedema and pleural effusion~
  ICD9CM Diagnosis Code: 428.1
```

The title of the report has been conveyed in the observation identifier (OBX.3 field 00571) as a coded entry. The body of the report has been encoded in the observation value (OBX.5 field 00573) using the HL7 TX data type.<sup>25</sup> The TX data type specifies that leading spaces are significant and that paragraphs are specified as repeating fields. Therefore “hard carriage returns” between paragraphs are signaled not by new line characters, or any other control character, but rather by whatever was specified in the MSH segment as the repeat delimiter, usually ‘~’. The example has been shown as multiple lines to fit neatly on the page, but no such new lines appear in the encoded message. It would really look more like:

```
OBX||TX|^Chest X-Ray Report||Description of procedure.~-
  PA, lateral views were obtained~Findings.~- Blunting of the
  left costo-phrenic angle, cardiomegaly and interstitial
  lines~Conclusions.~- Pulmonary oedema and pleural
  effusion~ICD9CM Diagnosis Code: 428.1|r
```

Any occurrence of delimiter characters in the text need to be preceded by the appropriate escape character, usually ‘\’.

In HL7, the entire report does not need to be transmitted as a single OBX segment. A limited structure can be imposed by sending an ordered list of OBX segments preceded by an OBR segment that acts as a “header,” and conveys the name of the report as the universal service ID (OBR.4 field 00238).<sup>26</sup> For example, the same report could be encoded instead as:

```
OBR||||^Chest X-Ray Report|...r
OBX|1|TX|^Description of procedure||PA, lateral views were
  obtained|r
OBX|2|TX|^Findings||Blunting of the left costo-phrenic
  angle, cardiomegaly and interstitial lines|r
OBX|3|TX|^Conclusions||Pulmonary oedema and pleural
  effusion|r
```

24. In this example some liberty has been taken with the transcoding into plain text to make it more readable, such as eliding concept names for findings that are redundant with the headings, and renaming the heading for the ICD9 code.

25. Another alternative would be to use the formatted text (FT) data type, which allows for greater control over formatting but requires extensive use embedded formatting commands like “\.sp\” to end a line and skip one vertical line, and so on.

26. This model is designed to convey a “battery” of tests, each test as individual OBX observations, with the name of the battery specified in the OBR segment.

```
OBX|4|CE|^Diagnosis||428.1^^I9|\r
```

This approach allows the top level headings to be conveyed as coded entries by using the observation identifier field (OBX.3 field 00571), which always has a CE data type. In the case of the last observation, a CE data type rather than a TX data type has also been used for the observation value (OBX.5 field 00573). This mechanism is very similar to the DICOM SR approach to encoding a report that uses coded concept names for the first level of headings. Unfortunately, the same mechanism can not be used to recursively encode nested sub-headings. A DICOM SR that uses containers with concept names to represent nested subheadings will need to be converted into plain text, at least below the first level of headings.

Also notice how the set ID (OBX.1 field 00569) is used to convey the sequence of the individual observations.

The OBR and OBX segments are combined with other segments (such as PID to identify the patient) into messages such as an ORU (unsolicited transmission of an observation) message.<sup>27</sup>

References to objects (such as images) that are not embedded in the report can be encoded in an HL7 OBX segment by using the reference pointer (RP) data type. For example, a reference to a DICOM image might be sent as:

```
OBX||RP|^BaseLine||
IMAGE99^&1.2.840.113619.6.48.99&ISO^Image^DICOM|\r
```

The reference allows for a system dependent text “pointer” to identify the object. This pointer is not unique across systems. In addition, an application ID can be encoded with the same form as a hierarchic designator (HD) data type. This allows for a unique identifier to be conveyed.<sup>28</sup> The “Image/DICOM” type of data and sub-type are defined in HL7 to specify an image encoded with a PS 3.10 File Meta Information header.<sup>29</sup>



## HL7 Clinical Document Architecture

Recognizing that there is a need to exchange structured documents within a health-care environment, HL7 has begun to define a Clinical Document Architecture (CDA).<sup>30</sup> The CDA was previously known as the Patient Record Architecture (PRA).

27. There are other similar messages, such as those that are a response to a query.

28. It is not clear from reading the HL7 standard whether or not this application identifier is intended to uniquely identify the object itself, or the application that is responsible for the object. To reference DICOM objects, this seems to be the only place that it makes sense to convey the DICOM SOP Instance UID.

29. The type is also specifically constrained to images. Quoting from HL7 V2.3.1 page 2-48: “In HL7, the use of DICOM data is limited to images only.” It is not clear whether this restriction is a political (turf) concern, or merely meant to convey that a different type than “Image” should be used when a referenced DICOM object is not an image. Types and sub-type combinations like “Audio/DICOM” and “Application/DICOM” would seem to make sense. Do not confuse the RP type and sub-types with MIME content types and sub-types, since they are not identical.

The choices of the original name and the new name are telling. On the one hand, the original name implied that the intent of the effort was to standardize the encoding of the electronic patient record. On the other hand, the new name would seem to reflect the reality that defining the structure and encoding of a document is not sufficient to create an electronic patient record, but is a useful goal in its own right.

The term “architecture” is not used in the sense of a system architecture but rather refers to the organization of the CDA framework itself. The so-called “level one” CDA, which is what has been defined so far, specifies the mechanism for structuring and encoding a document. In future, the architecture will also contain “level two” and “level three” specifications, which will constrain the allowable structures and semantics based on document type code, and specify the markup of clinical content, respectively. By a crude analogy with DICOM structured reporting, CDA level one is equivalent to the definitions of the SR modules, level two is equivalent to the definition of the SR IODs and SOP classes, and level three is equivalent to the definition of templates.

The encoding of the CDA document<sup>31</sup> is in XML. The constraints on the form of the document are specified in an XML Data Type Definition (DTD)<sup>32</sup> and accompanying prose.<sup>33</sup> The DTD for the “header” of a CDA document is derived in a formal manner from the HL7 V3 Reference Information Model (RIM). In CDA Level 1, the “body” of the document is not as strictly constrained. Before going on to describe the CDA in more detail, it is necessary to first consider XML itself.

## XML: Beyond the Hype

The eXtensible Markup Language (XML) is variously touted as a panacea for all forms of data interchange problems, the universal basis for all future file formats or the ultimate format for getting information in and out of databases. XML is really just yet another form of low level encoding, one layer above ASCII plain text. Indeed it is less powerful (without augmentation) than ASN.1 for describing data structures of even moderate complexity<sup>34</sup>.

Where XML excels is in the encoding of text based hierarchical structures, where each node in the hierarchy is identified by an element, encoded as an <element> and

---

30. The information in this section is based on the version 1.0 draft for membership ballot, dated 4 August 2000.

31. The expression “CDA document,” as used in the HL7 proposal, is an awkward one, since when spelled out it becomes “Clinical Document Architecture document.”

32. A DTD in XML is a set of rules that define the syntax that is allowed to be used in the document. If you like, it is analogous to a “grammar” for the document. It specifies which elements and attributes may or shall occur, with what multiplicity, in what order, and with what nesting.

33. The intention is that an XML Schema will be used to more formally define the constraints, when the W3C has finalized the Schema recommendation.

34. ASN.1 (Abstract Syntax Notation 1) is used to specify amongst other things ISO OSI protocols, and is paired with a set of Encoding Rules (such Basic Encoding Rules (BER)) to specify how to turn data structures into bytes (octets). It is also used to encode X.509 certificates.

</element> pair. It is human readable and editable which has great appeal for toy applications. Like any of the “\*ML” family, it is derived from the Standard Generalized Markup Language (SGML), a more complex standard that is widely used in the document publishing industry.

Like XML, DICOM structured reporting is all about encoding hierarchical structures of information. It is necessary to consider how structured reports might be exchanged with applications that exist outside the DICOM world, where XML may play a role. There are also valuable lessons to be learned from work that extends XML to support transformations and style sheets (such as eXtensible Stylesheet Language - Transformations (XSL-T)), as well as references to internal and external content (XPath, XPointer and XLink).

## What is XML?

XML looks very similar to HTML, derived as they both are from SGML. As such, the appearance of XML is very familiar to the hordes of individuals who have tried their hand at creating web pages by typing HTML text. What distinguishes XML is that tags have no predefined meaning as they do in HTML, and imply no particular presentation. While a <BOLD> tag in HTML means that the browser should render text in a bold font style, a <BOLD> tag in XML means whatever one wants it to mean, perhaps an attribute of someone’s personality. In other words, XML encodes only structure, with no implied semantics. It needs to be combined with a dictionary of meanings for tags for it to be useful.

The fact that XML documents are typed as plain text makes them very friendly to those who need to generate or interpret them. Unfortunately this also means that hand generated XML, or XML created programmatically using print statements, runs the risk of being badly formed or invalid, as is commonly the case with HTML pages. To use XML robustly requires the use of tools that parse and write XML based on some internal representation or API. Such tools are available and in widespread use, but if one is not going to massage the document by hand, one may well ask if there is significant advantage to a text rather than binary encoding.

Still and all, few people are comfortable reading hexadecimal dumps of binary encoded data like DICOM. Text based formats seem to give people a warm fuzzy feeling and reassure them that they are in control.

For example, what follows is a hex dump of a fragment of a DICOM structured report:

```

                                FE FF 00 E0 FF FF FF 40 00
10 A0 43 53 10 00 48 41 53 20 4F 42 53 20 43 4F
4E 54 45 58 54 20 40 00 40 A0 43 53 06 00 50 4E
41 4D 45 20 40 00 43 A0 53 51 00 00 FF FF FF FF
FE FF 00 E0 FF FF FF FF 08 00 00 01 53 48 06 00
30 30 30 35 35 35 08 00 02 01 53 48 06 00 4C 4E
63 65 6D 6F 08 00 04 01 4C 4F 12 00 52 65 63 6F
72 64 69 6E 67 20 30 62 73 65 72 76 65 72 FE FF
0D E0 00 00 00 00 FE FF DD E0 00 00 00 00 40 00
```

```
23 A1 50 4E 10 00 53 6D 69 74 68 5E 4A 6F 68 6E
5E 5E 44 72 5E 20 FE FF 0D 0E 00 00 00 00
```

The equivalent meaning when dumped by a DICOM parsing tool looks something like this:

```
(ffff,e000) Item
(0040,a010) Relationship Type <HAS OBS CONTEXT>
(0040,a040) Value Type <PNAME >
(0040,a043) Concept Name Code Sequence
  (ffff,e000) Item
  (0008,0100) Code Value <209069>
  (0008,0102) Coding Scheme Designator <99PMP>
  (0008,0104) Code Meaning <Observer>
  (ffff,e00d) Item Delimitation Item
(ffff,e0dd) Sequence Delimitation Item
(0040,a123) Person Name <Smith^John^^Dr^>
(ffff,e00d) Item Delimitation Item
```

By contrast, one very literal XML encoding of the same information might look something like this:

```
<contentitem>
  <contentlabel>1.1</contentlabel>
  <relationshipType>HAS OBS CONTEXT</relationshipType>
  <conceptname>
    <codesequence>
      <codevalue>209069</codevalue>
      <codingschemedesignator>99PMP
      </codingschemedesignator>
      <codemeaning>Observer
      </codemeaning>
    </codesequence>
  </conceptname>
  <valuetype>PNAME</valuetype>
  <personname>Smith^John^^Dr^</personname>
</contentitem>
```

It is clear that the XML example is more readable to a novice. Even the use of “familiar” words to identify elements like “<concept name>” seems attractive compared to intimidating hexadecimal numbers for group and element tags like “(0040,A043).”<sup>35</sup>

To a program, it makes little difference whether it has to parse binary data or text data to read the same structure. The inefficiencies of lexically analyzing text, skipping white space, and so on have long since been dwarfed by the enormous processing power of modern computers, and one can always compress the text if concerned about the size of the data set.

---

35. Though in practice XML element names very quickly lose their intuitiveness and readability in real applications rather than toy examples, especially when name spaces are used.

## Alternative XML Encoding Issues

Since one is free to define the meanings of XML tags, someone has to decide which tags to use in order to make a document interchangeable. Conversely, if the XML form is being used internally, then the choice of tags makes no difference, as long as they are consistent within a single application. The issue of whether or not there should be a standard set of tags defined by some organization for structured reports will be deferred until later in the chapter.

For now, some of the alternative forms for encoding SR constructs will be examined, by elaborating on the example. In the first pass at this, every attribute of a DICOM encoding of the SR construct is transcoded into an equivalent XML tag. So for example, there is a <conceptname> tag corresponding to the DICOM Concept Name Code Sequence attribute. Furthermore since it is a coded sequence, and coded sequences are so common, a specific <codesequence> tag is used to encapsulate the attributes of a coded sequence item. This allows one to use the DTD mechanism to specify that a <conceptname> tag contains a <codesequence> tag, rather than have to repeat the contents of a code sequence every time.<sup>36</sup> The DTD doesn't really have a macro mechanism per se, and this expedient keeps things tidy at the cost of bloating the nesting of the tags.

Note especially that the notion of a coded entry has been retained. This means that elements like concept names will always be indirected through an external dictionary. An alternative would be to define a specific XML tag for every equivalent code that might be used. For example one could define an <Observer> tag,<sup>37</sup> and instead of encoding:

```
<codesequence>
  <codevalue>209069</codevalue>
  <codingschemedesignator>99PMP
</codingschemedesignator>
  <codemeaning>0bserver</codemeaning>
</codesequence>
```

just write:

```
<Observer>
```

This looks very neat, but it would mean that an entire “shadow dictionary” of XML tags equivalent to all the coded entries that might be used would need to be defined.

Perhaps one could also elide the use of the PNAME value for the <valuetype> tag, and just put the person's name directly between the <Observer> and </Observer> tags, specifying somewhere<sup>38</sup> that invalid values will not be used here:

36. Although XML “entities” can be used (like macros) to factor out commonality in a DTD without creating new elements for this purpose.

37. For clarity, in these examples the XML name space capability has not been used. Name spaces would almost certainly be used in practice to disambiguate element names defined by different organizations.

38. Perhaps by using the proposed XML Schema mechanisms to describe limits on character data.

```

<contentitem>
  <contentlabel>1.1</contentlabel>
  <relationshipType>HAS OBS CONTEXT</relationshipType>
  <Observer>Smith^John^Dr^</Observer>
</contentitem>

```

One might also want to change the relationship from being included content of the <contentitem> tag to an attribute of that tag instead, and do the same for the <contentlabel> tag:<sup>39</sup>

```

<contentitem
  contentlabel="1.1"
  relationshipType="HAS OBS CONTEXT">
  <Observer>Smith^John^Dr^</Observer>
</contentitem>

```

Another alternative would be to elide the explicit encoding of a content item altogether, and make everything either the content or an attribute of a <Observer> tag:

```

<Observer
  contentlabel="1.1"
  relationshipType="HAS OBS CONTEXT">
  Smith^John^Dr^
</Observer>

```

This certainly appears very attractive in terms of compactness and clarity. However, it does mean that for every tag like <Observer> one would need to define rules in the DTD or XML Schema about what attributes are required to be present, what their potential values are and so on.<sup>40</sup> This would allow use of off-the-shelf validating tools that can verify the compliance of the XML syntax with the specified DTD or Schema.

## A Standard XML SR Encoding?

By now it should be clear that while XML can be used to encode a DICOM structured report, there are a plethora of alternatives and trade-offs involved. These include:

- ▶ whether or not to use explicit tags for every concept and code, or use a coded entry mechanism
- ▶ what to specify as attributes of a tag and what to include as nested content
- ▶ how to encode SR constructions that have no equivalent in XML, such as relationships and value types

39. The content label was used explicitly here for clarity, even though it is implicitly defined in the DICOM encoding. XML doesn't "natively" have internal pointer mechanisms, but can be extended using specifications like XLink, XPointer and XPath to use explicit labels and pointers.

40. The decision as to when to use an element or an attribute in XML is an interesting one. Attributes have stronger typing than elements, in that there are primitive types for tokens and identifiers as well as character data. Attributes are particularly attractive for identifiers like the labels for content items.

- ▶ how to specify the form of values beyond the parsed character (#PCDATA) that XML is limited to<sup>41</sup>
- ▶ how to transcode binary data into text<sup>42</sup>
- ▶ how to transcode the various forms of internationalized text used in DICOM to the Unicode used in XML<sup>43</sup>
- ▶ how to make these decisions to maximize the use of off-the-shelf parsing and validating tools, specifying structure and constraints in a machine readable formal syntax

One of the strengths of XML is the opportunity to specifying structure and constraints declaratively (using DTDs and XML Schema) rather than procedurally. The expectation is that such tools and specification languages will be more powerful and more readily available than domain specific tools, such as those developed for DICOM.<sup>44</sup>

However, in the absence of such tools, the risk of using XML as an interchange format<sup>45</sup> is that documents may be encoded that do not comply with the SR definitions and constraints.

Inevitably, there have been calls to transcode DICOM SR into XML for the purpose of interchange. However, a great deal of time and effort would need to go into developing the definition of any such transcoding and it is not clear what the benefits would be. So far, there has been considerable reluctance on the part of DICOM Committee to take on such an effort. Sufficeth to say there has been lively debate on the matter. The current policy is

- ▶ to defer the specification of a standard SR XML encoding for interchange purposes until there is a need in the IS world; specifically to delegate the task to the

---

41. In future, it is hoped that XML Schema will provide a stronger means of typing data, but for now “de facto” or application specific conventions are required, which cannot make use of off-the-shelf validation tools, requires consistent behavior between documents, and increases the risk of creation of incorrect but valid and well-formed XML documents.

42. This is less of an issue for SR than other DICOM objects, since almost no SR constructs use binary data, except the spatial coordinate floating point values, which can easily be converted to decimal strings. For objects with bulk binary data such as images and waveforms, it essentially precludes by value XML encoding.

43. This is not strictly necessary, since one can specify localized encodings at the start of an XML document. However, since all XML parsers are required to support UTF-8 and UTF-16, this makes little sense from an interoperability perspective. The default, when no encoding is explicitly specified, is UTF-8. Obviously, common localized encodings like “us-ascii” and “iso-8859-1” are likely to be widespread in addition to the Unicode encodings. Since most XML handling is performed in Java, which uses Unicode internally and provides conversions for most of the common character repertoires, and most programmers use off-the-shelf XML parsers, these issues will be transparent in many cases. Recent versions of PERL also support UTF-8.

44. A common complaint about DICOM is that it is not specified in a formal syntax that is machine readable, and from which parsers and validators can automatically be generated.

45. As opposed to a purely internal, proprietary representation in XML, such as an intermediate form as the input to a style sheet driven transformation engine.

joint DICOM-HL7 working group and to perform the work in the context of the proposed HL7 CDA (Clinical Document Architecture) Level 3 definition<sup>46</sup>

- ▶ not to standardize the tags or conventions for the use of XML for internal and presentation purposes

The rationale behind this strategy is simply that, in the DICOM world, DICOM SR encoding is complete and sufficient. For internal purposes such as presentation, the use of XML tools such as XSL-T and XML-XSL aware browsers can be tremendously useful, but no standard is necessary for this. Beyond the DICOM world, other standards bodies such as HL7 are planning to use both a model similar to SR and XML as components of their new architecture. If any group felt sufficiently strongly that the SR model needed to be encoded in XML independently of DICOM or HL7, there would be nothing to prevent them from establishing an entirely new organization to do the work.

## XML and Presentation

Though the foregoing may seem relentlessly negative, there is one place in which leveraging off-the-shelf XML tools which do already exist makes a lot of sense. That is in the area of presentation of DICOM SR documents.

Like a DICOM SR document, an XML document, even with a DTD, specifies absolutely nothing about how it is to be presented. In this sense, an XML document encodes only meaning.

However, when combined with a “style sheet” of some kind, a tool or application can create a presentation. Two standards for style sheets that are applicable to XML documents and relatively mature are Cascading Style Sheets (CSS) and eXtensible Stylesheet Language (XSL).<sup>47</sup> By using a style sheet tailored to a particular DTD, an XML document can readily be converted to something presentable like HTML or PDF. In particular, one can expect new web browsers to be able to render the combination of an XML document and an appropriate style sheet. Obviously generic browsers can do little with XML documents by themselves, other than display the element names and character data.

A key part of this approach is that the style sheet, regardless of the standard form in which it is encoded, must be appropriate to the type of document. Whether the form of the XML document is specified by its DTD, or more precisely in future by an XML Schema, the style sheet must specify how to render particular tags or patterns of

---

46. Level 1 of CDA has passed ballot, but specifies only the header and a broad outline of the content. Until more specific content markup is defined in CDA Level 3, transcoding DICOM SR into CDA involves loss of fidelity.

47. One will often see references to XSL-T rather than XSL alone. The “T” stands for “transformation.” There are really two parts to XSL, the part that specifies the transformation rules, which is familiar to most people, and the “formatting objects” (FO) which provide more precise control over layout but are still in development and are in the “ambitious” category. XSL also depends on the XPath specification to identify what elements and attributes are to be “matched” to trigger invocation of a transformation rule.

tags. Indeed style sheets consist of a set of rules that are successively applied looking for matching patterns, and specifying what to do when a match occurs. In most cases what is done is to rewrite the tree into another tag-based presentation-oriented format such as HTML.

The role of pattern transformation languages in general and XSL-T in particular will be examined further in the chapter on “Trees, Traversal and Transformation.”

## HL7 CDA: Structure and Use of XML

One can see from the foregoing that XML has many potential roles associated with the use of DICOM Structure Reporting. One such role is in the encoding of HL7 CDA documents, which are defined to be in XML.

The CDA is not defined in a vacuum. It makes use of the proposed version 3 data types, as well as the proposed version 3 Reference Information Model (RIM). However, each instance of a CDA document is self-contained. The specification includes examples of how such documents may be encapsulated in MIME messages and transmitted with in version 2.x OBX segments.

There are no security features in the CDA per se, but privacy and digital signature concerns can be addressed by the encapsulating message transmission layer.<sup>48</sup> An ability to tag either the entire document or selected segments with a “confidentiality status” is provided.<sup>49</sup> There is no similar feature in SR.

A CDA document consists of a header and a body. These are divided in much the same manner as a DICOM structured report is split into administrative information in the general modules and content information in the document content module. Like SR, the body is a tree of nested content. Unlike SR, the tree consists of only containers, which may be one of four types: sections, paragraphs, lists and tables. There are no explicit relationships. Within a container there may be content that is plain text, links to external documents or multimedia, and coded entries. The CDA addresses the issue of presentation in addition to semantic content, to a much greater extent than SR. For example, it allows for formatted text as well as “local markup.”

Like SR, the CDA anticipates extensive use of controlled terminology in coded entries. Unlike SR, the use of coded entries for concept names is not mandatory. For example, a CDA “section” may have a “caption.” The caption may be plain text and may or may not have a coded entry. The vocabulary domain for the codes for captions is defined to be from LOINC, though other coding schemes may also be used.<sup>50</sup> Paragraphs and items (in lists) may also have captions.

---

48. *Creating digital signatures for XML documents is a non-trivial problem. Entity substitution, as well as white space and comment removal, make it difficult to maintain a bit-preserved document to sign. Various “canonical” representations for the purpose of computing a reproducible message authentication code (MAC) have been proposed. See the book “XML and Java” for further discussion.*

49. *Though obviously, without a cryptographic approach, an application can choose to ignore the confidentiality status.*

The four basic CDA containers of sections, paragraphs, lists and tables have been mentioned. These distinctions are not present in DICOM SR, which has only one type of container.<sup>51</sup> In CDA, paragraphs and lists can occur within sections, list items or table cells. Items occur only within lists. Within such containers, “content” occurs. Specifically, content occurs in table cells, paragraphs, list items, and nested within other content.<sup>52</sup> In addition to recursively containing further content, content elements may contain plain character data, links, coded entries, observation media or local mark-up. Unlike in SR, apart from references and coded entries, the CDA does not distinguish content type further: it is all plain text.<sup>53</sup> Despite the plethora of specific data types proposed in HL7 V3, these are not used in the CDA body at the content level, other than for coded entries (though they are used extensively in the header).

Tables basically follow the XHTML model, and have headings, rows and columns, defined by <th>, <tr> and <td> elements respectively. Table headings may have coded entries along the lines of captions. The existing DICOM SR framework has no equivalent concept, though any table can be encoded as a restricted form of a tree. What is presently lacking in SR is a template that specifies a consistent pattern for encoding a table.

The language of any character data in a CDA document may be specified explicitly.<sup>54</sup> Though SR allows for the possibility of doing a similar thing, the concept modifiers required for this have not yet been standardized. Note that specifying the language is different from specifying the character set. Both CDA and SR have an ability to specify the character set in use, through the generic mechanisms provided by XML and basic DICOM encoding respectively.<sup>55</sup>

“Links” may be specified either to external content that is referenced or to locations inside the same document. External links are distinguished from “observation media” which are pointers to external multi-media content that is an integral part of the report. DICOM SR does not make such a distinction. Like SR, CDA does not include multi-media content “in-line.” Though the ED data type and MIME Base64 encoding could be used for such a purpose, HL7 is apparently waiting for other standards on embedding binary content in XML documents.

---

50. This is so because the data type for the coded caption is specified to be “Coded, With Extensions” (CWE) rather than “Coded, No Extensions” (CNE).

51. Though one could argue that the “continuity of content” flag in an SR container effectively creates two different sub-classes of the CONTAINER value type.

52. Content may also occur in local mark-up.

53. Or, in XML-speak, “PCDATA” which stands for “parsed character data.”

54. By using the values from RFC 1766 in <xml:lang> attributes, as defined in the XML 1.0 Recommendation.

55. The character set in use is defined at the start of an XML document. All XML parsers are required to support both Unicode UTF-8 and UCS-16, though most often one will see US-ASCII or ISO-8859-1. In DICOM, the Specific Character Set attribute is used with nationally specific character sets. DICOM does not support Unicode, hence during transcoding, character set translation must be considered. See also the chapter on “Internationalization and Localization.”

The ability to reference internal links is potentially equivalent to SR's by-reference relationship mechanism. That is, both CDA and SR potentially have the ability to encode a structure that is more complex than a tree. The encoding of links in CDA is currently modeled after the HTML anchor tag.<sup>56</sup>

## CDA Coded Entries

Earlier, when considering how to transcode DICOM structured reports into XML, the question was raised as to how to mark up coded entries. The same issues of how to use XML elements and attributes arises for the CDA. The choices that have been made by HL7 provide guidance for anyone transcoding DICOM SR into XML.

The CDA uses an element `<coded_entry>` which contains, in addition to an optional identifier and local markup, a single `<coded_entry.value>`. The value does not contain plain character data, but rather is defined in terms of attributes. These attributes include the mandatory value "V" and scheme "S" (encoded as an ISO OID<sup>57</sup>), as well as optional components such as a scheme name "SN" and a display meaning "DN".<sup>58</sup> For example:

```
<coded_entry>
  <coded_entry.value
    V="T-04000"
    S="2.16.840.1.113883.6.5"
    SN="SNM"
    DN="Breast"
  />
</coded_entry>
```

Coded entries can be used anywhere that other content can be used. They can even reference the "original text that supports the assigned code," as in the following example:

```
<content>
  The mass in the upper outer quadrant of the right
  <content ID="x1234">breast</content> measures 3 cm
```

---

56. The intent is expressed in the proposal that when the W3C finalizes a recommendation for pointers and links it will be adopted into Level 1 CDA.

57. The following OIDs for coding schemes are mentioned in the current CDA draft:

2.16.840.1.113883.6.5 SNOMED

2.16.840.1.113883.6.3 ICD9-CM

2.16.840.1.113883.6.1 LOINC

There is not yet any provision for using the "old" (HL7 2.x) style of specifying of coding schemes as string values rather than OIDs (though they can optionally be conveyed in the SN attribute). However, this matter is currently under review, and may change before the V3 Data Types are balloted.

58. This description corresponds to the definition of the V3 data type of Concept Descriptor (CD) as proposed in "Version 3 Data types," Ballot Draft 1, Revision 1.2, 2000/08/01, and the corresponding XML representation described in "XML Implementable Technology Specification (ITS) for V3 Data types," Ballot Draft.

```
in diameter.
<coded_entry>
  <coded_entry.value
    ORIGTXT="x1234"
    V="T-04000"
    S="2.16.840.1.113883.6.5"
    SN="SNM"
    DN="Breast"
  />
</coded_entry>
</content>
```

The notion of text that “supports an assigned code” is an interesting one. There is no equivalent of this idea in SR, though one could use the concept modifier relationship to achieve the same effect.

## Transcoding CDA and SR

Just as it is possible to transcode a complex DICOM structured report into plain text for transmission, printing or display, so it is possible to convert an SR into a CDA Level 1 document.<sup>59</sup> No doubt the inverse transformation is also possible. The issue however, is how to transcode documents between the two standards with as little loss of fidelity and functionality as possible. In this section, specific issues that arise during transcoding will be discussed.

Coded entries in their simplest form are a relatively easy matter. Though the HL7 V3 data type insists in an ISO OID coding scheme designator, a one-to-one mapping exists between string designators and their OID equivalents. The code meaning (display name) is optional in HL7, but a dictionary of terms can be consulted during a conversion to DICOM. Many of the attributes for the enhanced mode of encoding are not supported by the V3 CD data type, but additional XML attributes could be added to the <coded\_entry.value> attribute as extensions. Transcoding of the CDA “original text that supports the assigned code” can be achieved by using an appropriate concept modifier relationship in SR.

Mapping of SR containers to CDA elements could be performed at either the section, paragraph or list item level, using the SR concept name as a CDA coded entry caption. During the inverse transformation, from CDA to SR, unless the type of container was preserved in the DICOM container concept name (perhaps as a modifier), the distinction of container type would be lost.

DICOM SR by-reference relationships and CDA links to internal elements are essentially equivalent, though in CDA one cannot specify the type of relationship. External references in CDA documents (either as links or observation media) can be repre-

---

*59. In future, if appropriate SR constructs are incorporated in the HL7 RIM, then CDA Level 3 documents may be able to support a more faithful translation. It is the intent of DICOM to work with HL7 to achieve this goal.*

sented as DICOM UID, composite, image or waveform references, only once any non-DICOM objects so-referenced have been converted into DICOM equivalents.

Serious difficulties arises with the explicit relationships and value types in SR, including coordinates. For relationships, one can insert in a CDA document an additional layer of nesting of containers, using a coded entry for a caption that is equivalent to an SR relationship. An example of a “has properties” relationship in SR might be:

```
<CODE:(,,“Finding”)=(,,“Mass”)>
  <has properties CODE:(,,“Shape”)=(,,“Round”)>
```

If one ignores the relationship, a simplistic translation into CDA might look like:<sup>60</sup>

```
<paragraph>
  <caption>
    <caption_cd V="" S="" DN="Finding"/>
  </caption>
  <content>
    <coded_entry><coded_entry.value
      V="" S="" DN="Mass"/>
    </coded_entry>
    <paragraph>
      <caption>
        <caption_cd V="" S="" DN="Shape"/>
      </caption>
      <content>
        <coded_entry><coded_entry.value
          V="" S="" DN="Round"/>
        </coded_entry>
      </content>
    </paragraph>
  </content>
</paragraph>
```

In this example, the “shape” child of the “mass” finding has been encoded as a nested paragraph. Notice that the “name-value” pair concept of SR is not explicit in the CDA translation. Here, content that is not a nested container is assumed to be the “value” of the node, and the caption<sup>61</sup> is used as the “name.”

The relationship might be explicitly encoded as the caption of an intervening paragraph (with no content) as follows:

```
<paragraph>
  <caption>
    <caption_cd V="" S="" DN="Finding"/>
```

<sup>60</sup>. Note that one can not leave the V and S attributes blank like this, but to reduce the complexity of the example, actual attribute values have been elided for everything except the display name. That is, an actual coded caption might be ‘<caption\_cd V=“209001” S=“1.2.840.113619.6.48” SN=“99PMP” DN=“Finding”/>’.

<sup>61</sup>. Notice that the form for specifying a coded entry for a caption is different from a coded entry for content. There is a specific “<caption\_cd>” that is used.

```

</caption>
<content>
  <coded_entry><coded_entry.value
    V="" S="" DN="Mass"/>
  </coded_entry>
  <paragraph>
    <caption>
      <caption_cd V="" S="" DN="has properties"/>
    </caption>
    <paragraph>
      <caption>
        <caption_cd V="" S="" DN="Shape"/>
      </caption>
      <content>
        <coded_entry><coded_entry.value
          V="" S="" DN="Round"/>
        </coded_entry>
      </content>
    </paragraph>
  </paragraph>
</content>
</paragraph>

```

Value types (in the body of the document) also present a problem. In SR, there are specific value types for various atomic types such as names of people, dates and times, unique identifiers and so on. The numeric measurement value type also has an explicit form which includes a coded entry for units. While these certainly can be converted into plain character data for use in a CDA Level 1 document,<sup>62</sup> there is no guarantee that such a conversion will be possible in the opposite direction.

For example, an SR content item with a numeric value type such as:

```
<NUM:(,, "Diameter")="1.3" (,, "cm")>
```

might be converted into a CDA equivalent like:

```

<content>
  <caption>
    <caption_cd V="" S="" DN="Diameter"/>
  </caption>
  1.3
  <coded_entry><coded_entry.value
    V="" S="" DN="cm"/>
  </coded_entry>
</content>

```

In this conversion, the units are distinguished only by the order of their appearance. Alternatively, one might use a specific coded caption for a container such as a paragraph to describe the units, as in the following:

<sup>62</sup>. Hopefully this problem will be rectified in CDA Level 3.

```

<paragraph>
  <content>
    <caption>
      <caption_cd V="" S="" DN="Diameter"/>
    </caption>
  </content>
  1.3
  <paragraph>
    <caption>
      <caption_cd V="" S="" DN="Units"/>
    </caption>
    <content>
      <coded_entry><coded_entry.value
        V="" S="" DN="cm"/>
      </coded_entry>
    </content>
  </paragraph>
</paragraph>

```

Notice how additional containers are necessary, since content elements may not include containers such as paragraphs or sections. Given the need for an additional outer container, one might as well tag that with a caption that specifies the value type as well:

```

<paragraph>
  <caption>
    <caption_cd V="" S="" DN="Numeric Value"/>
  </caption>
  <content>
    <caption>
      <caption_cd V="" S="" DN="Diameter"/>
    </caption>
    ...
  </content>
  ...
</paragraph>

```

The point of this example is that there are many alternatives, each of which preserves a greater or lesser amount of the information in the source document. When confronted with a CDA document that was once an SR document and has been converted previously, an application attempting to recover the SR content will likely have some difficulty.<sup>63</sup> Such a “round-trip” conversion is a difficult test of the fidelity of a transcoding design. Even more difficult is the common case, that of a unidirectional conversion where the original content was created without any expectation of conversion. It is likely to be just as difficult to create sections, paragraphs, lists and tables when converting an SR document to a CDA document, as it is to identify numeric

<sup>63</sup> It is possible to hide SR specific features in “local markup” (privately defined XML elements) which is flagged with attributes to indicate that it is not to be rendered. This hidden information could potentially be recovered during a round-trip conversion.

measurements, person names, dates and times buried in CDA plain character data for conversion to SR value types.

Spatial and temporal coordinates are probably an intractable problem for conversion until further CDA data types are defined. The CDA Level 1 has no such data type. In a conversion from SR to CDA the best that can be done is to apply the coordinates to the referenced images and waveforms and replace the coordinates with references to the pre-formatted rendered equivalents.<sup>64</sup> The problem does not arise in the opposite direction, since CDA Level 1 documents will never contain coordinates.<sup>65</sup>

As these examples illustrate, transcoding with high fidelity is not easy.

Since there is no possibility (short of encapsulation) of preserving the bits in a transcoded document, any digital signatures will be invalidated. Some proxy mechanism for re-signing a previously signed source document would be necessary, as described in the chapter on “Security: Privacy and Authenticity.”

An open question that remains for any system that uses a non-literal representation of a report, is the legal standing of the documents. Anything less than an exact image of the document actually used (for some purpose) is open to criticism, since it might not be possible to reproduce it “exactly” subsequently. There is a continuum of reproducibility from paper documents through pre-formatted text (such as PDF), word-processing formats, presentation-oriented mark-up to semantic mark-up like CDA and DICOM SR. A likely scenario is that reports will be created by a radiologist in a DICOM world, converted into CDA for distribution, translated into HTML and finally rendered in a referring physician’s web browser. The fidelity of this chain of conversions may well be questioned, since what was created by the radiologist may differ “significantly” from what the referring physician ultimately sees.

**Not yet  
standardized.  
Final standard  
may differ  
substantially.**

## Clinical Observations Access Service

It remains to discuss the role of the Clinical Observations Access Service (COAS)<sup>66</sup> defined by the Object Management Group (OMG).<sup>67</sup>

Underlying COAS is the CORBA standard itself, which provides for the distributed consistency of objects throughout a network. Multiple applications on multiple nodes can access the same objects. One may consider CORBA as a non-proprietary,<sup>68</sup> object-oriented extension of the various “remote procedure call” standards that were

64. That is, create an image with a “burned in” annotation that illustrates the reference to the temporal or spatial region of interest.

65. Except perhaps in a “round-trip” conversion as mentioned earlier, in which some private convention will no doubt be used to “hide” the coordinates inside the CDA document.

66. The version of COAS referenced here is the 1999/03/25 Final Submission.

67. Or more specifically, defined by the OMG’s CORBAMED group. CORBA stands for Common Object Request Broker Architecture.

68. The only significant vendor not involved in CORBA is Microsoft, who have their competing DCOM architecture. Regardless, adoption of CORBA has been hampered by its complexity, as well as the popularity of Java Remote Method Invocation (RMI) which can do some of the same things.

popular in the late eighties and early nineties of the last century. CORBA is usually classified as “middle-ware,” that is it has a role in the middle, between the business logic of an application and the operating system, platform, database and network specific code. The basic CORBA standard provides services for accessing and modifying objects in a distributed manner using standard protocols. The APIs are specified in a language-independent Interface Definition Language (IDL).<sup>69</sup> Beyond CORBA itself, the OMG defines services for general purposes, such as those that support transactions, security, naming, and so on. In addition, “vertical” domains are addressed by specific task forces. CORBAMed is one such task force.

As the name suggests, COAS addresses the matter of “access” to “observations.” It defines an interface that allows for queries, for browsing, for consumers and suppliers of events, as well as for asynchronous and federated access.<sup>70</sup> The conformance mechanism is defined in terms of which of these interfaces a provider supports. The bulk of the COAS submission addresses the interfaces for accessing observations, and is beyond the scope of this book.<sup>71</sup>

The best way to get a sense of how COAS may be useful is to turn to the back of the submission and study the appendices that describe usage patterns, scenarios and client implementation examples. Of particular interest is the suggestion to supply XML templates as “input qualifiers” to filter responses. Reading this example also raises a potential difficulty with conformance and interoperability, if the provider does not support XML documents as input qualifiers.

It is interesting, however, to examine in more detail the classes of observations used in COAS and to compare them with what is available in SR. COAS observations may be “compound” or “atomic.” An atomic observation is a single object with an associated value. This classification is reminiscent of the HL7 V2.x concepts of individual tests as OBX observations and batteries of tests as an ordered sequence of OBX segments preceded by an OBR “header.” Each observation has a “type,” which in COAS means the name of the observation (i.e. it is equivalent to an SR concept name or a CDA caption).

Observation values may be of various kinds which include:

- ▶ coded element
- ▶ loosely coded element
- ▶ plain text
- ▶ measurement
- ▶ date-time
- ▶ curve

---

*69. Indeed IDL is an enduring product of the OMG which is used in many non-CORBA applications. For example, the W3C Document Object Model (DOM) is specified in IDL.*

*70. The federated access interface is for use by legacy systems that can “push” information but cannot be queried.*

*71. And, frankly, beyond the author’s expertise.*

- ▶ multi-media
- ▶ technology instance locator
- ▶ no information

Coded elements are defined by using the “qualified code” class.<sup>72</sup> A qualified code consists simply of a coding scheme identifier and a code. Where it is used in an observation it may also be accompanied by a plain text meaning.

Loosely coded elements are an interesting idea. The notion is that where a code would normally be sent, if the lexicon available does not contain the necessary code, a plain text value can be sent instead. That is, the code set can be “extended” with plain text informally and on demand. In SR, this is essentially equivalent to a template allowing either a code or text value type for a concept.

For plain text, a language may be explicitly specified.

Curves simply provide X and Y points and coded definitions of the axes. It is interesting that neither DICOM SR nor the HL7 CDA provide such a data type.

Rather than re-inventing ways to define multi-media content, in COAS MIME Content Types are used. Interestingly for a standard like CORBA that supports the transfer of binary information, the multimedia payload is conveyed with a MIME encoding.<sup>73</sup>

A “technology instance locator” is a generalization of the concept of a URL.<sup>74</sup> The idea is that an observation may be specified by an http or ftp URL, for example.

The date-time observation is interesting in that it is also accompanied by relational operators (such as “equal to” or “less than”) as well as specifiers of accuracy (such as “within” and “plus or minus”).

Measurements have values specified as a single numeric value or as a ratio. They may have optional (coded) units, as well as ranges (upper and lower bounds) and precision specified.<sup>75</sup> An array of values can also be specified as a time-series.

The “no information” observation is an explicit place-holder for missing information. It may be accompanied by a coded explanation such as “not asked” or “not available.”

Relationships between observations can be indicated by references that are associated with a coded reference type. The set of relationships is extensible. The set described in COAS is derived from the CEN Electronic Health-care Record Communication -

---

72. The qualified code object is not actually defined in COAS, but rather in an earlier CORBAMED specification, the Lexicon Query Service (LQS).

73. It might have been more appropriate to separate the specification of content type from the delivery of the payload.

74. Uniform Resource Locator.

75. When the DICOM SR numeric value type was being defined, similar features were suggested. It was decided that it was better to encode these as additional nodes in the SR tree rather than encumber the basic encoding. It remains for templates to be defined that add such features to SR.

Part 2: Domain Term List.<sup>76</sup> The COAS submission describes conventions for constructing unique qualified names strings using a hierarchical mechanism. For example:

```
"DNS:omg.org/DS0bservationAccess/relation/CENTC251N98116/
IsReportedBy"
```

Observations can have modifiers described by observation “qualifiers” which are also conveyed by codes. The set described in COAS is derived from HL7 V2.3 field names, but mention is made of using terms from the same CEN document that was the source of codes for relationships.

## Summary

It should be clear by now that, when it comes to encoding structured documents, there is more than one way to “skin a cat,” so as to speak. This chapter has dwelt on the differences between approaches, particular those that are a barrier to bidirectional transcoding. It is also reassuring to notice the similarities however.

Some of the key points mentioned in this chapter include:

- ▶ DICOM structured reports are not the only possible encoding of structured health-care documents
- ▶ HL7 version 2.x is in widespread use and provides a mechanism for transmitting atomic observations in OBX segments, or a list observations in OBX segments preceded by an OBR header segment
- ▶ the name of an OBX segment is a coded entry and the value may be of various types, include coded entries, plain text, formatted text and encapsulated data (ED)
- ▶ non-HL7 data can either be encapsulated or transcoded for inclusion in an HL7 message
- ▶ a binary object such as a DICOM encoded SR can be Base64 encoded and transmitted in an HL7 2.x OBX segment as ED
- ▶ a package of objects can be encapsulated in a MIME package (as they would be in an e-mail attachment or the body of an http message), and sent in an HL7 2.x OBX segment
- ▶ with transmission using HL7 version 2.x, transcoding of DICOM structured reports is confined to use of a single level of headings and conversion into plain text
- ▶ the HL7 Clinical Document Architecture (CDA) defines a structured document “format”
- ▶ there are many similarities between SR and the CDA, but it is particularly difficult to transcode relationships as well as value types other than coded entries and plain text
- ▶ the CDA does not possess the concept of temporal or spatial coordinates, so these are likely to be lost in any transcoding

---

76. Specifically, from an earlier draft of CEN preENV 13606-2:2000.

- ▶ the CDA is encoded in XML, and the choices made as to the use of elements and attributes provide guidance for those transcoding DICOM SR into XML encoding for other purposes
- ▶ at the present time, the DICOM organization has expressed an intent not to define a formalized encoding of SR in XML
- ▶ the OMG Clinical Observations Access Service (COAS) mostly dwells on the definition of interfaces for accessing observations, but does contain classes that define atomic and compound observations with associated codes and types

# ***Trees, Traversal and Transformation***

---

By now it should be abundantly clear that information in a DICOM structured report is contained in a tree of content items. This tree is encoded in a DICOM data set as recursively-nested variable-length sequence items. The “external” or “serialized” DICOM representation is intended for interchange and storage purposes. It is not normally a suitable representation for “internal” use within an application in which the tree needs to be traversed, searched, manipulated or transformed.

This chapter is about implementation. It will address the subject of how the content tree may be represented. A traditional data structure will be compared with object model and event stream approaches. The relevance of W3C recommendations and de facto standards like the Document Object Model (DOM) and Simple API for XML (SAX) will be discussed. A proposal to define a Structured Reporting Object Model (SROM) based on DOM will be described.

In addition, the subject of rule based pattern matching and transformation will be considered, particularly with reference to converting the semantic information contained in a structured report into something presentable. The role of eXtensible Stylesheet Language Transformations (XSL-T) in specifying such transformations will be described.

## **Tree Data Structures**

Turning the calendar back a decade or more, when presented with a requirement to encode an SR tree, a programmer of the last century might have written something like this in conventional C:<sup>1</sup>

---

1. For simplicity's sake, the concept name and value are represented as abstract data types “struct coded\_entry” without further elaboration. Further details are not relevant to this discussion. Relationships have also been elided.

```
struct node {
    struct node *nextsibling;
    struct node *firstchild;
    struct coded_entry name;
    struct coded_entry value;
};

struct node *root;
```

Such a simplistic representation is sufficient to both generate a tree in a top-down manner, and to parse it similarly. In this context, “top-down” means to begin from the root node. Eliding checks for various errors, the creator of a tree might write:

```
root=(struct node *)malloc(sizeof(node));
root->nextsibling=null;
root->firstchild=null;
/* ... fill in name and value ... */

child=(struct node *)malloc(sizeof(node));
child->nextsibling=null;
child->firstchild=null;
/* ... fill in name and value ... */

root->firstchild=child;
```

and so on. Given the root node, traversing the tree becomes a simple matter of following the forward linked lists of siblings and children, stopping when a null pointer is encountered.

If one needed to walk backwards to previous siblings or ancestors, one could provide appropriate links in each node:

```
struct node {
    ...
    struct node *previoussibling;
    struct node *parent;
    ...
};
```

Simple as such an approach might appear, few people write code this way anymore. Rather, object-oriented methods are used to separate the details of the internal representation from the logic of the application (i.e. encapsulation).

## Object Model

Stepping back and looking at the SR tree from an application’s point of view, what are the requirements for an internal model of the tree? For the creator of a tree, facilities must be provided to

- ▶ create a root node
- ▶ create and add sibling and child nodes
- ▶ assign names, values and relationships to nodes

The user of a tree must be able to:

- ▶ find the root node
- ▶ find the siblings and children of a node
- ▶ access names, values and relationships of nodes

The editor of a tree must also be able to:

- ▶ detach a node and its descendants (a “sub-tree”) from a tree
- ▶ insert a node as a sibling of a specified node, or between nodes
- ▶ insert a node as a child of a specified node

and so on. The point is that these facilities are generally applicable to any tree, not just an SR tree. Furthermore, even those facilities that are specific to SR, such as the names and values, are independent of the internal representation. By defining classes that represent the facilities in an object-oriented manner, one can take advantage of

- ▶ encapsulation, by hiding the internal representation of the tree (such as whether it is in memory or on disk, and whether it uses pointers to dynamically allocated nodes, pre-allocated tables, a database of nodes or whatever)
- ▶ inheritance, by defining base or abstract classes that represent a tree in a generic sense, specializing the classes as necessary (for example a base tree class, an SR tree class, an SR tree class that can be serialized to disk, etc.)
- ▶ polymorphism, by defining common operations that apply to any tree node regardless of its actual class (such as getNextSibling(), getFirstChild() and so on)

This approach is obviously more desirable than having every function in an application that needs to create a node allocate memory, fill in pointers, and so on.

For example, were one creating such classes from scratch in C++ one might write:

```
class node {
    class node *getNextSibling(void);
    class node *getFirstChild(void);
};
class srnode : public node {
    coded_entry getName(void) const;
    coded_entry getValue(void) const;
};
class srnode_mutable : public srnode {
    void setName(coded_entry name);
    void setValue(coded_entry value);
};
```

and so on. Ideally, one would not write this all from scratch, but would reuse some suitable existing library of tree-handling code.

Regardless, the application that is creating, using or manipulating the SR tree need have no knowledge of the internal representation of the tree.<sup>2</sup> The object-oriented approach is more desirable than the data-structural approach, but one can do better still in terms of reuse and portability, by taking a language independent approach.

## Document Object Model

As has been mentioned before, a DICOM SR tree is not that different from an XML tree. Both DICOM SR and XML have as their primary structure recursively-nested content. Though the details differ, perhaps XML tools can be reused in the context of DICOM SR. In particular, the Document Object Model (DOM) may be of use.

DOM is a W3C recommendation that is very popular amongst XML developers. The goal of DOM is to provide language and representation independent access to the content of an XML document.<sup>3</sup> This is achieved by:

- ▶ defining an object that is a “document” that may be read from and be written to an external representation
- ▶ defining methods that may be used to traverse the document, and examine and insert “nodes” that are its content
- ▶ hiding any details of the internal representation of the document
- ▶ defining the objects and methods in a language-independent Interface Definition Language (IDL)<sup>4</sup>
- ▶ providing multiple language “bindings” so that the interface can be implemented in widely-used languages<sup>5</sup>

Though the details of DOM are not directly relevant, a few simple examples of its use will illustrate the power of the concept. An XML document may be read, parsed and validated as follows:<sup>6</sup>

```
Parser parser = new Parser("dummy");
Document document = parser.readStream(xmlInputStream);
if (parser.getNumberOfErrors() > 0) return;
```

Having “de-serialized” the XML document from the input stream into an opaque internal representation, one can traverse the document tree from the top downwards, extracting and examining nodes, moving on to their siblings and descendants. For example:

```
for (Node
    node=document.getDocumentElement().getFirstChild();
    node != null;
    node=node.getNextSibling()) {
    ...
}
```

2. In the most extreme cases, it could even be “located” in a database on another machine entirely. The “business logic” of the application needn’t know and shouldn’t care.
3. Another goal is to validate an XML document against its DTD, but that role will not be discussed here.
4. Specifically the Object Management Group’s (OMG) IDL.
5. Currently standard bindings are defined for Java and ECMAScript.
6. This Java example uses the specific objects and methods of the IBM XML4J implementation. The examples are derived from Maruyama et al. “XML and Java: Developing Web Applications.” Addison-Wesley 1999. ISBN 0-201-48543-5. This is a brief, readable, inspiring and indispensable book for anyone interested in XML, DOM and SAX.

3

and so on. Already details of XML that affect the design of DOM are creeping into the examples. For instance, an XML document doesn't have a single root node as a DICOM SR document does, hence the iteration on siblings. The model of the XML document distinguishes between "elements" and "nodes," there being many different types of nodes that are not elements, such as attributes, character data, and so on.

How is one to make best use of DOM in the context of SR, given that there are many DOM features specific to the peculiarities of XML, and many SR peculiarities not supported directly by DOM? Possibilities include:

- ▶ transcoding a DICOM SR document into XML, then using DOM
- ▶ parsing a DICOM SR document directly into an internal representation and implementing a DOM interface, according to a defined DICOM to XML transformation, but without ever actually instantiating the XML representation
- ▶ use the ideas from DOM to define a DICOM SR specific object model, reusing the same objects and methods where appropriate, adding additional objects and methods to support SR specific constructs

The first approach, transcoding into XML, is feasible though not necessarily desirable, as is discussed in the chapter on "Other Standards: HL7, MIME, XML, COAS."

The second approach, "simulating" an XML document but directly parsing the DICOM encoding sounds attractive at first. In reality, it suffers from the worst of both worlds: existing XML to DOM parsing tools cannot be used, since a DICOM specific parser is necessary, and the fit of DOM constructs to represent SR is poor in the first place.<sup>7</sup>

The best approach seems to be to learn from DOM, but implement a specific object model that applies to DICOM SR. This approach avoids slavishly following XML conventions that are not directly applicable, and allows the full fidelity of the SR document to be represented cleanly.

## SR Object Model

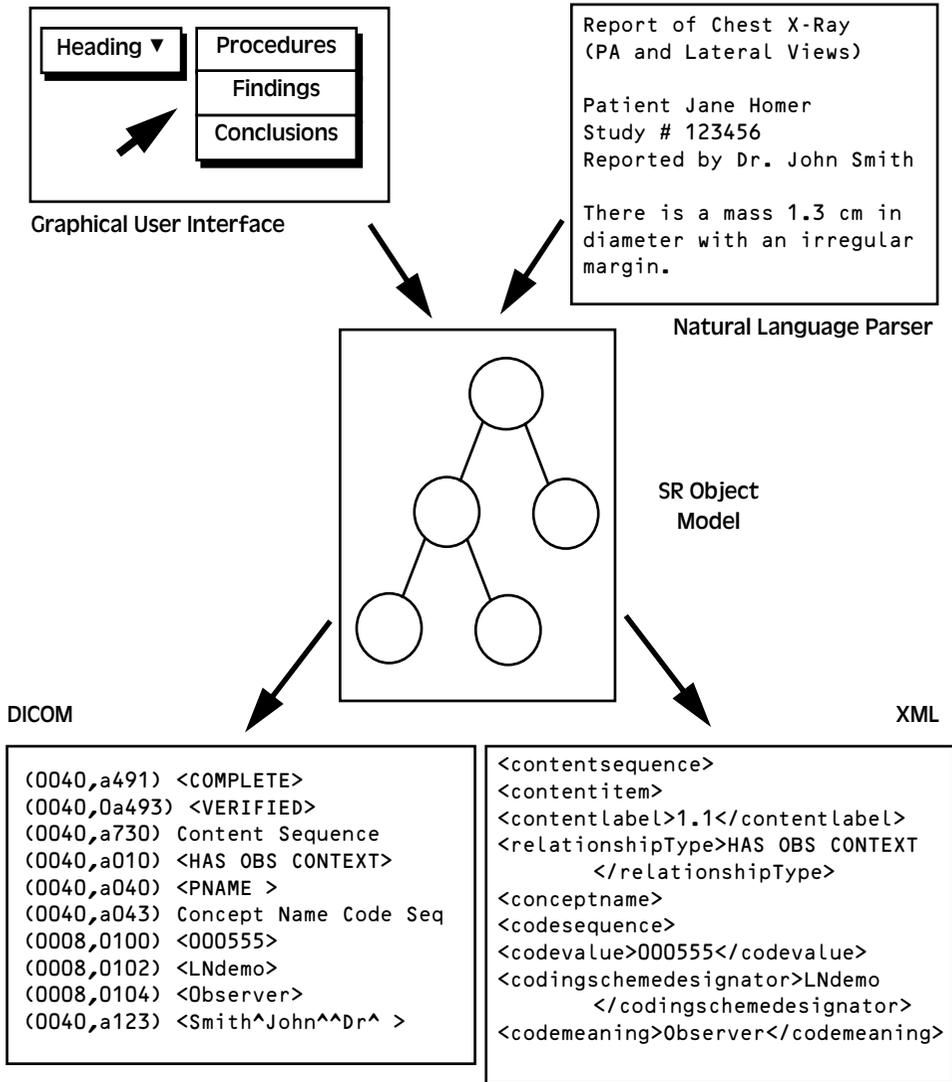
DICOM is not in the business of defining APIs<sup>8</sup> for programmers, but that does not mean that there is no opportunity for consensus amongst implementers. As DOM and SAX have proven in the XML world, there can exist a clearly defined boundary between tools that encode and parse documents, and tools that read or manipulate information content. The value of defining such a boundary cannot be underestimated.

For example, the talent, skills and resources necessary to design and build toolkits for "low level" DICOM encoding are totally different from those necessary to construct

---

7. The "simulated" XML document approach does have merit under other circumstances however, and will be revisited in the discussion of SAX and XSL-T.

8. Application Program Interface(s).



**FIGURE 51. SR Object Model: Input Applications**

the “business logic” in applications that understand the meaning of the content. Furthermore, by separating these two groups across a common API, one may also achieve independence from the representation of the data. One may quibble ad infinitum about whether the DICOM encoding of an SR document or a hypothetical XML encoding of the same content is “better” by some criterion. However, if parsers for both are implemented but accessed through a common API, then the “users” of the information need never concern themselves with the external representation.

The same argument applies to rendering SR documents for human consumption. The expertise in creating “rendering engines” can be made independent of the serialized encoding of the source document through the use of a common object model inter-

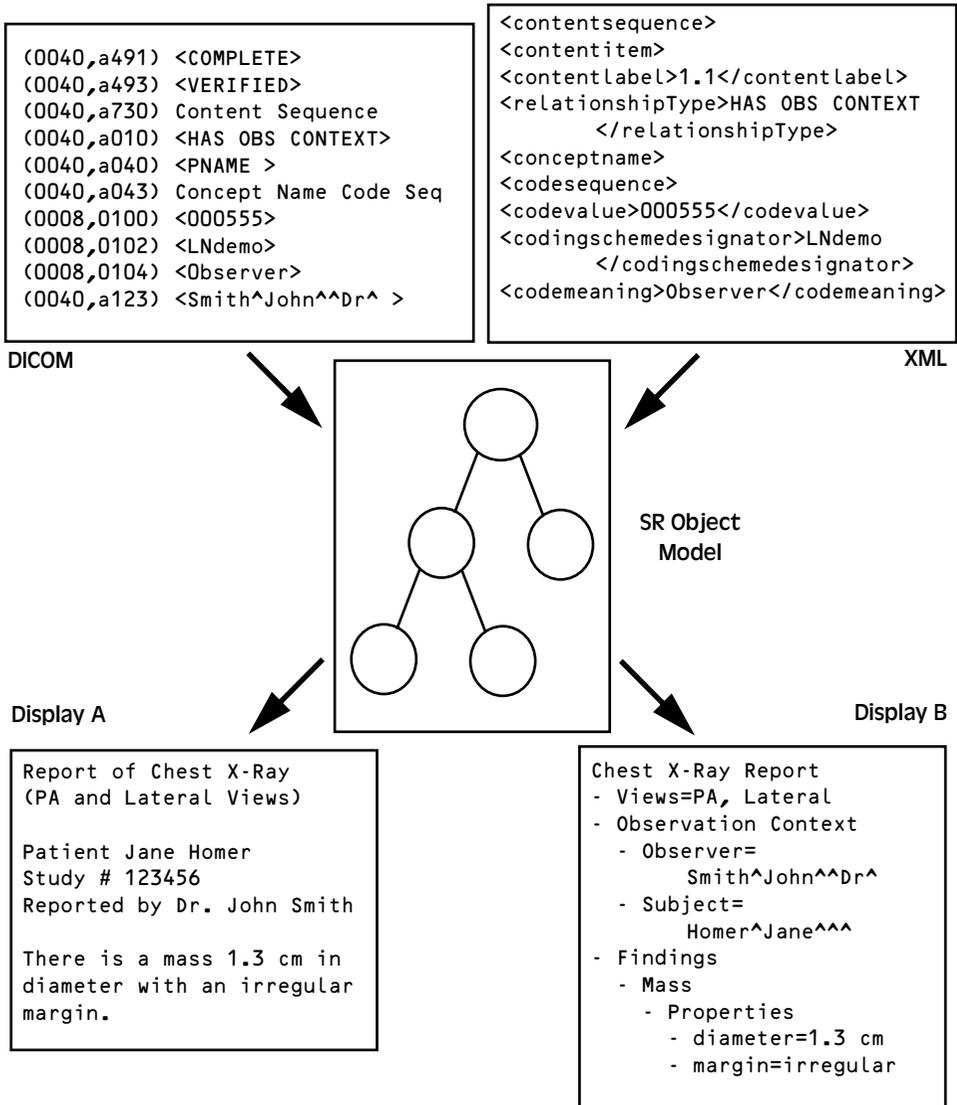


FIGURE 52. SR Object Model: Rendering Applications

nally. Equally, transformation tools that “rewrite” the tree can also be made independent.

The requirements for the object model in general were described earlier. The SR object model differs from the XML DOM in that:

- ▶ SR nodes are name-value pairs
- ▶ DOM nodes may be elements, attributes or plain character data<sup>9</sup>
- ▶ SR nodes have explicit relationships between parent and child, that may be modeled as attributes of the child node

- ▶ DOM has no equivalent concept to relationships; the only relationship is containment and it is implicit
- ▶ SR node names are coded entries and values may be of various types
- ▶ DOM has no coded entry mechanism for element names, nor does it have a repertoire of value types<sup>10</sup>

Taking these differences into account, and using the DOM (Core) Level 1 interfaces as a guide, one can define an SR-specific API that reuses as many DOM interfaces and methods as possible. In particular, the Node interface can be adopted. The generic Node interface can be extended from DOM to include:

- ▶ a coded entry concept name
- ▶ a value type
- ▶ a relationship type

Specific sub-classes of Node can be created to correspond to the SR-specific value types.<sup>11</sup>

The generic traversal and editing methods can be reused. These include such methods as Node.getFirstChild(), Node.getNextSibling(), Node.insertBefore() and so on. In addition, since nodes are named (by their concept name), it is also appropriate to add additional methods that operate not as iterators but as accessors to specific named nodes.<sup>12</sup> For example, a method such as Node.getFirstChildNamed(name) is potentially extremely useful.<sup>13</sup>

Table 18 and Table 19 summarize the interfaces and methods that might be useful for an SR Object Model. A more complete description with language bindings is contained in the annex on the “Structured Reporting Object Model.”

---

9. There are other types of DOM node, such as entities entity references, processing instructions.

10. This reflects the limits of the XML DTD framework; there is no means to specify the type of text in element values; attributes have a greater range of types but these have specific purposes.

11. DOM doesn't actually sub-class Node, but rather has enumerated node types which determine which methods are usable.

12. The NamedNodeMap in DOM has a similar purpose, and is used for attributes, However, it is not inherited from NodeList, it does not preserve the order of its contents, and it cannot contain two entries with the same name.

13. One may well ask why the libraries of iterators supplied in modern languages are not used. The object model is intended to be language neutral, and have multiple language bindings. Accordingly, specific methods are defined rather than using, say, the Java 2 Collections or the C++ Standard Template Library.

TABLE 18. SR Object Model - Interfaces

Interface	Extends	Contains	SR Entity
CodedEntry			Code Sequence macro
Node		CodedEntry concept name relationship type value type	Content item
ContainerNode	Node	continuity of content	CONTAINER
CodeNode	Node	CodedEntry value	CODE
TextNode	Node	string value	TEXT
NumericNode	Node	string value, CodedEntry units	NUM
PNameNode	Node	string value	PNAME
DateNode	Node	string value	DATE
TimeNode	Node	string value	TIME
DateTimeNode	Node	string value	DATETIME
UIDNode	Node	string value	UIDREF
CompositeNode	Node	object reference	COMPOSITE
ImageNode	CompositeNode	object reference	IMAGE
WaveformNode	CompositeNode	object reference	WAVEFORM
SCoordNode	Node	coordinates and type	SCOORD
TCoordNode	Node	coordinates and type	TCOORD
ReferenceNode	Node	reference to another node	By-reference relationship

## Using an SR Object Model

A few examples of how the methods of class Node might be used will illustrate the approach.<sup>14</sup> Initially traversal of an existing tree begins at the root node. The `getFirstChild()` and `getNextSibling()` methods may be used to iterate through the first level of descendants as follows:

```
void iterate(Node root) {
    for (Node child=root.getFirstChild(); child != null;
        child=child.getNextSibling()) { // process child }
}
```

A recursive traversal is a straightforward extension of the same pattern. One needs to check each child to be sure it is not a leaf before following its descendants:

```
void process(Node node) {
    // pre-order traversal actions would go here
    if (node.hasChildNodes()) {
```

14. Most of the examples in this section follow the patterns described in "XML and Java."

```
        for (Node child=node.getFirstChild();
             child != null; child=child.getNextSibling()) {
            process(child);
        }
    }
    // post-order traversal actions would go here
}
```

Creating a brand new SR object model from scratch involves first creating a root node:<sup>15</sup>

```
Node root = (Node)nf.createContainerNode(
    cf.createCodedEntry("209068", "99PMP", "Report"),
    null, "SEPARATE");
```

Notice that one always creates a specific type of node that is a sub-class of a generic node. In the case of a container node, a coded entry concept name may be specified, and the continuity of content needs to be specified explicitly. Since this node is the root node, there is no relationship with a parent node, hence the relationship is specified as null.

Descendants can then be created and appended as in the following example:

```
root.appendChild(nf.createTextNode(
    cf.createCodedEntry("209007", "99PMP", "Procedure"),
    "CONTAINS", "PA and lateral"));
root.appendChild(nf.createTextNode(
    cf.createCodedEntry("209001", "99PMP", "Finding"),
    "CONTAINS", "Multiple masses"));
```

In the case of nodes that are descendants of other nodes, the relationship with any potential parent node must be specified when the node is created, because it cannot be changed later.<sup>16</sup>

---

15. In the examples in this section, there will be assumed to be an instance node factory that creates nodes, "nf", and a coded entry factory that creates coded entries, "cf". In SROM, as in DOM, the Factory pattern is used, rather than using constructors, so the "new" operator is not used. In DOM the factories for all objects is contained in the Document class (the method is Document.createElement()). In SROM, there are separate factories for nodes and coded entries. The factories are instances of implementation specific classes that implement the SROM NodeFactory and CodedEntryFactory interfaces.

16. Nodes are essentially immutable. While they may be moved inserted, deleted or replaced, their attributes cannot be changed, i.e. there are no "set" methods. In particular, there is no Node.setRelationshipType() method. One could argue that the Node.appendChild() and similar methods should perhaps take a relationship type as an argument, to replace any existing relationship type defined in the node being added.

TABLE 19. SR Object Model - Methods for Interface Node

Returns	Method	Parameters	Exceptions	DOM equivalent
CodedEntry	getNodeName()			yes
String	getNodeType()			yes
String	getRelationshipType()			no
Node	getParentNode()			yes
Node	getFirstChild()			yes
Node	getFirstNamedChild()	CodedEntry name		no
Node	getLastChild()			yes
Node	getPreviousSibling()			yes
Node	getNextSibling()			yes
Node	getNextNamedSibling()	CodedEntry name		no
Node	insertBefore()	Node new, Node ref	SRROMException	yes
Node	replaceChild()	Node new, Node old	SRROMException	yes
Node	removeChild()	Node old	SRROMException	yes
Node	appendChild()	Node new	SRROMException	yes
boolean	hasChildNodes()			yes
Node	cloneNode()	boolean deep		yes

An existing tree of nodes can be modified by using the `insertBefore()`, `removeChild()`, `replaceChild()` and `appendChild()` methods. When using these methods, note that:

- ▶ `insertBefore(new,ref)` adds the “new” child before the existing “ref” child; if “ref” is null then the “new” child is added as the last child,<sup>17</sup> i.e. is equivalent to `appendChild(new)`; if “ref” is not null, it must be an existing child
- ▶ `replaceChild(new,old)` replaces the “old” child with the “new” child; neither may be null and “old” must be an existing child
- ▶ `removeChild(old)` removes the “old” child; “old” must be an existing child

A special case that needs to be considered is how to intermingle methods that iterate and modify the tree of nodes. In the case of removing nodes while iterating through a list of children, one must keep removing from the head of the list rather than following the successors of a node that has already been removed (and hence whose successors will have been set to null). In other words, this works:

```
while (node.hasChildNodes())
    node.removeChild(node.getFirstChild());
```

and this does not (it only removes the first node):

```
for (Node child=node.getFirstChild(); child != null;
     child=child.getNextSibling()) node.removeChild(child);
```

17. That is, as the last child of the Node whose `insertBefore()` method is being called.

## Iterators with Names

The SR object model extends the iterators described in DOM to include versions that make use of the name of a node (or more specifically, the coded entry that represents the concept name). The CodedEntry class has an equals() method that compares the code value and coding scheme designator of two nodes to determine equality, ignoring the contents of code meaning. This equals() method is used by the getFirstNamedChild() and getNextNamedSibling() iterators.<sup>18</sup> The following example creates an SR document with a root node and several children, and then iterates through it to find a named child:

```
Node root = nf.createContainerNode(
    cf.createCodedEntry("209068", "99PMP", "Report"),
    null, "SEPARATE");
root.appendChild(nf.createTextNode(
    cf.createCodedEntry("209007", "99PMP", "Procedure"),
    "CONTAINS", "PA and lateral"));
root.appendChild(nf.createTextNode(
    cf.createCodedEntry("209001", "99PMP", "Finding"),
    "CONTAINS", "Multiple masses"));
root.appendChild(nf.createTextNode(
    cf.createCodedEntry("209005", "99PMP", "Conclusion"),
    "CONTAINS", "Metastases"));

Node finding=root.getFirstNamedChild(
    cf.createCodedEntry("209001", "99PMP", ""));
```

In this example, getFirstNamedChild() will return the node that is the “finding” of “multiple masses.” Notice that the coded entry for “finding” will be matched, even though the instance of the CodedEntry used is not the same instance, and the code meaning is different.

As a final example, consider a routine that searches an entire SR tree starting from the root, locating all occurrences of findings:

```
void search(Node node, CodedEntry wanted) {
    if (node.hasChildNodes()) {
        for (Node child=node.getFirstChild();
            child != null; child=child.getNextSibling()) {
            search(child, wanted);
        }
    }
    if (node.getNodeName().equals(wanted)) {
        /* found ... do something deep and meaningful */
    }
}

CodedEntry finding =
```

18. That is, equality is defined by the value of the CodedEntry, not by the same reference to the same instance of a CodedEntry object.

```
cf.createCodedEntry("209001","99PMP","");
search(root,finding);
```

Notice in this example that since one has to visit each node anyway, in order to recursively search its descendants, the iterators-by-name are not used. Also notice that a “depth-first” traversal is defined, since the check for a match and subsequent processing are performed *after* recursively visiting all of a node’s descendants. The matching and processing code can be placed differently to achieve other traversal orders.

Since the test for equality is based solely on the code value and coding scheme designator, the code meaning may be localized or internationalized, or even absent, without invalidating the search. In this example, it was left as an empty string in the search key.

## An Alternative Approach - Event Streams

Both DOM and SROM are aimed at hiding (encapsulating) the internal representation of a tree by providing methods for traversing and manipulating the structure. Though not strictly necessary,<sup>19</sup> there is a presumption that the tree lives in memory. That is, an external document has been parsed into an internal representation, and perhaps validated during the process of parsing, or as a separate step. Alternatively, the tree may have been created *de novo*. After traversal or manipulation, the internal representation may be used directly, say to drive a display application, or it may be “serialized” into an external representation, such as an XML or DICOM SR document.

An alternative approach is to consider the top-down traversal of an XML or SR document tree as being a stream of events. In both forms of document, content is recursively nested. It is easy to detect the beginning and the end of a particular structure of content.

For example, during the process of a parsing a fragment of an SR document of the form:

```
<contains CONTAINER:(,,“Description of procedure”)>
  <contains TEXT:(,,“Description”)="PA, lateral views">
```

events of the following form might be generated:

```
startContainer:
relationship: contains
conceptname: (,,“Description of procedure”)
startText:
relationship: contains
conceptname: (,,“Description”)
textvalue: “PA, lateral views”
endText:
```

19. Conceivably, an implementation of the DOM or SROM interfaces could actually encapsulate an external representation, such as a database or a flat file that contained each node as a row in a table with indexed references between nodes.

endContainer:

This approach is used by another popular interface for handling XML documents, the Simple API for XML (SAX).<sup>20</sup> Suppose the SR fragment shown earlier was transcoded into XML in the following manner, using attributes for coded entries:

```
<container>
  <relationship type="contains"/>
  <conceptname>
    <codedentry ... DN="Description of procedure"/>
  </conceptname>
  <text>
    <relationship type="contains"/>
    <conceptname>
      <codedentry ... DN="Description"/>
    </conceptname>
    PA, lateral views
  </text>
</container>
```

The SAX events that would be generated during the parsing of such a fragment might look something like this:<sup>21</sup>

```
startElement: container
startElement: relationship
attribute: type="contains"
endElement: relationship
startElement: conceptname
startElement: codedentry
attribute: DN="Description of procedure"
endElement: conceptname
startElement: text
startElement: relationship
attribute: type="contains"
endElement: relationship
startElement: conceptname
startElement: codedentry
attribute: DN="Description"
endElement: conceptname
characters: "PA, lateral views"
endElement: text
endElement: container
```

In some respects, whether or not to use an “internal representation” or a “stream of events” is a matter of taste and style. However, the event-based approach potentially has the advantage that the size of the document that can be handled is not limited by available memory.<sup>22</sup>

---

20. SAX is not a product of any standards body or formal organization but a group of individuals. It is nonetheless both popular and well-defined. See also “<http://www.megginson.com/SAX>”.

21. Of course, these events are not rendered into text as stylistically represented here, but implemented as call-backs or by some similar mechanism.

## Transformation

There are many reasons why an SR document may need to be “transformed” into some other structure or representation. These include:

- ▶ transcoding to or from another standard representation
- ▶ transcoding to or from a proprietary representation
- ▶ transforming into a form for suitable presentation
- ▶ transforming into a more simple or more complex form

For example, in order to distribute a DICOM SR document via an HIS interface, it might be necessary to convert the document into HL7 CDA, which is encoded as XML. An existing ASCII-encoded name-value pair measurement report from the serial port of an ultrasound device might need to be transformed into a DICOM structured report. Since SR documents are not directly “human readable,” they often need to be converted into a directly renderable form such as HTML or PDF, perhaps with the use of a “style sheet” appropriate to the output device. A natural language parsing application may take a plain text report in an SR document and extract containers and codes to produce a more complex structure report. A speech recognition application might use an audio waveform referenced in a structured report and convert it into a plain text SR document.

Some of these scenarios, such as natural language parsing and speech recognition, involve relatively complex processing of the input, and usually require dedicated applications. Such applications can benefit from the use of a standard input or output API (such as DOM, SROM or SAX), in order to separate the “business logic” from any serialized external representation. However, one thing that many of the less complex transformation applications have in common is that they are amenable to a “rule-based” approach.

The application of transformation rules to trees is nothing new. One early application that springs to mind is the use of patterns of rules for “peephole” optimization of the assembly code output by programming language compilers. What is new is the popularity of structured documents on the Internet coupled with transformation rules in the form of “style sheets.” Both HTML and XML essentially encode trees. Style sheets define source patterns to match “sub-trees” and target patterns with which to replace them. Default rules may be applied in absence of more specific matches. Rules are reapplied until there are no further matches. The two common style sheet languages are Cascading Style Sheets (CSS) and eXtensible Stylesheet Language (XSL). Only the latter will be considered here, since it is more general and more powerful.

One of the primary applications of XSL is to transform documents that contain only semantic information, such as an XML document, into something that is suitable for

---

22. *In this day and age of cheap RAM and virtual memory, memory consumption might not seem like a realistic concern. However, it can be a significant constraint in dedicated devices using real-time operating systems, such as ultrasound machines. Furthermore, report servers rendering a high volume of requests do better not to squander memory in order to improve responsiveness and support a greater load.*

presentation, such as an HTML page. This is made easier by the fact that both XML and HTML are derived from SGML and share the same basic constructs. However, the rules and patterns described in an XSL style sheet, and the XSL engine that applies them, are potentially independent of any external representation. This observation is the key to applying XSL in the context of another representation, such as to a DICOM-encoded SR document.<sup>23</sup> How this independence from representation can be achieved will be described in the next section.

## **XSL-T**

XSL consists of two parts. First is a language for specifying patterns to match and rules to apply to transform a source document. This is the XSL Transformation language, or XSL-T. This is defined in a W3C recommendation. The other part of XSL defines a standard way in which output may be formatted. HTML has many inadequacies in terms of reproducibility of layout, and the proposed XSL Formatting Objects (XSL-FO) address this problem. Since the XSL-FO proposal is not complete, nor is it directly relevant here, formatting objects will not be considered further.

A brief introduction to the form of an XSL-T style sheet is in order. Suppose one had as a source document an XML encoded form of a DICOM SR fragment:

```
<text>
  PA, lateral views
</text>
```

and one wished to convert this into HTML of the following form for presentation:

```
<bold>PA, lateral views</bold><br>
```

To achieve this using XSL-T one might specify the following:

```
<xsl:template match="text">
  <bold><xsl:value-of select="."/;></bold><br>
</xsl:template>
```

One can see from this example that transformation rules may be defined as “templates.” The body of the template conveys what content to use to replace what was matched.

Access is provided not only to element values but also to attribute values, by using the features of the XPath specification on which XSL-T is predicated. For example, given the need to transform:

```
<codedentry V="" S="" DN="Description of procedure"/>
```

into

```
<p>Description of procedure</p>.<br>
```

one could use the following XSL-T fragment:

---

23. For other such insights, the reader is referred to an excellent book on the subject of XSL: Kay M. “XSLT Programmer’s Reference.” Wrox 2000. ISBN 1-861003-12-9.

```
<xsl:template match="codedentry">
  <p><xsl:value-of select="attribute::DN">
    </p><br>
</xsl:template>
```

In this case, the attribute referred to is of the “current” node. Alternatively, one could convert:

```
<code>
  <relationship type="has properties"/>
  <conceptname>
    <codedentry ... DN="Margin"/>
  </conceptname>
  <value>
    <codedentry ... DN="Irregular"/>
  </value>
</code>
```

into:

```
<p>has a <u>margin</u> which is <u>irregular</u></p>
```

by applying:

```
<xsl:template match="code">
  <p>has a
  <u><xsl:value-of select=
    "conceptname/codedentry/attribute::DN"></u>
  which is
  <u><xsl:value-of select=
    "value/codedentry/attribute::DN"></u>
  </p>
</xsl:template>
```

The intent here is not to describe the details of XSL-T, but rather to convey a sense of its power and flexibility. XSL-T allows for much more sophisticated application of patterns than simple repetition during recursive descent parsing, though the approach is strictly declarative rather than procedural.

## Using XSL-T with DICOM SR

XSL-T is also not the only approach to rule-based tree rewriting, and there is nothing to prevent one from defining one’s own rule language.<sup>24</sup> An argument could be made for defining a DICOM SR specific pattern language for this purpose. However, it makes sense, if possible, to reuse both tools that already available and expertise in their application.

How then, if structured reports are encoded in DICOM, can one make use of XSL-T? There are two approaches:

---

24. Indeed, the “XML and Java” book referred to earlier describes a simple bidirectional transformation language, LMX, and supplies an implementation using DOM.

- ▶ transform the DICOM SR into XML first, then apply XSL-T
- ▶ simulate XML that is equivalent to the DICOM SR constructs, and apply XSL-T

Both approaches require definition of XML equivalent for SR nodes. For example, to specify a “coded entry” to match with an XSL-T template, an appropriate XML element must be defined. However, in the second approach, the actual XML encoding need never be instantiated, if the XSL-T engine can accept either a DOM tree or a SAX event stream as input.<sup>25</sup>

For example, a DICOM SR parser that was given the following fragment:

```
<contains CONTAINER:(,,"Description of procedure")>
...
```

could generate events that simulated a hypothetical XML translation as follows:

```
startElement: container
startElement: relationship
attribute: type="contains"
endElement: relationship
startElement: conceptname
startElement: codedentry
attribute: V="" S="" DN="Description of procedure"
endElement: conceptname
...
endElement: container
```

The same approach can be used in the opposite direction if necessary. That is, the output of the transformation need never be instantiated as XML either, if an alternative form is required. Figure 53 illustrates various potential scenarios. Notice in particular that the output from the XSL-T engine can be any form of text: it does not need to be HTML or XML. The output could even be in the form of commands to a subsequent processing step to create a binary document such as a DICOM-encoded object. The XSL-T engine and the DOM and SAX interfaces act as boundaries between the inputs and the outputs ... any input scenario can potentially be used with any output scenario.

## Summary

Some of the key points mentioned in this chapter include:

- ▶ SR documents can be represented internally as a traditional tree data structure or as an “object model”
- ▶ there are advantages to standardizing the object model and separating it from the actual internal representation
- ▶ the W3C has defined a Document Object Model (DOM) for accessing internal representations of XML documents

---

<sup>25</sup>. As most popular implementations can. The approach has generally been to separate the XML parsing tool from the XSL-T transformation engine.

- ▶ a corresponding Structured Reporting Object Model (SROM) could be defined in order to support SR specific features that don't map well to XML
- ▶ an alternative approach to an object model is to parse a structured document into a stream of events that signal the beginning and end of components
- ▶ the Simple API for XML (SAX) is such an event model for XML
- ▶ either SAX or DOM can serve as a source of information for a rule-based tree transformation
- ▶ the eXtensible Stylesheet Language (XSL) is a language for defining such rules for transformations in a declarative, rather than procedural, manner
- ▶ XSL Transformation (XSL-T) engines can be driven by SAX events or DOM representations, and can produce any form of text-based output, including XML, HTML, PDF
- ▶ rule-based tree transformation can be used to both transform and create DICOM structured reports, as well as to convert to and from alternative representations such as HL7 CDA
- ▶ XSL-T rules may be used to create pre-formatted output in HTML or PDF from a simulated stream of XML events generated by a DICOM SR parser, without the need to ever actually instantiate an intermediate XML document

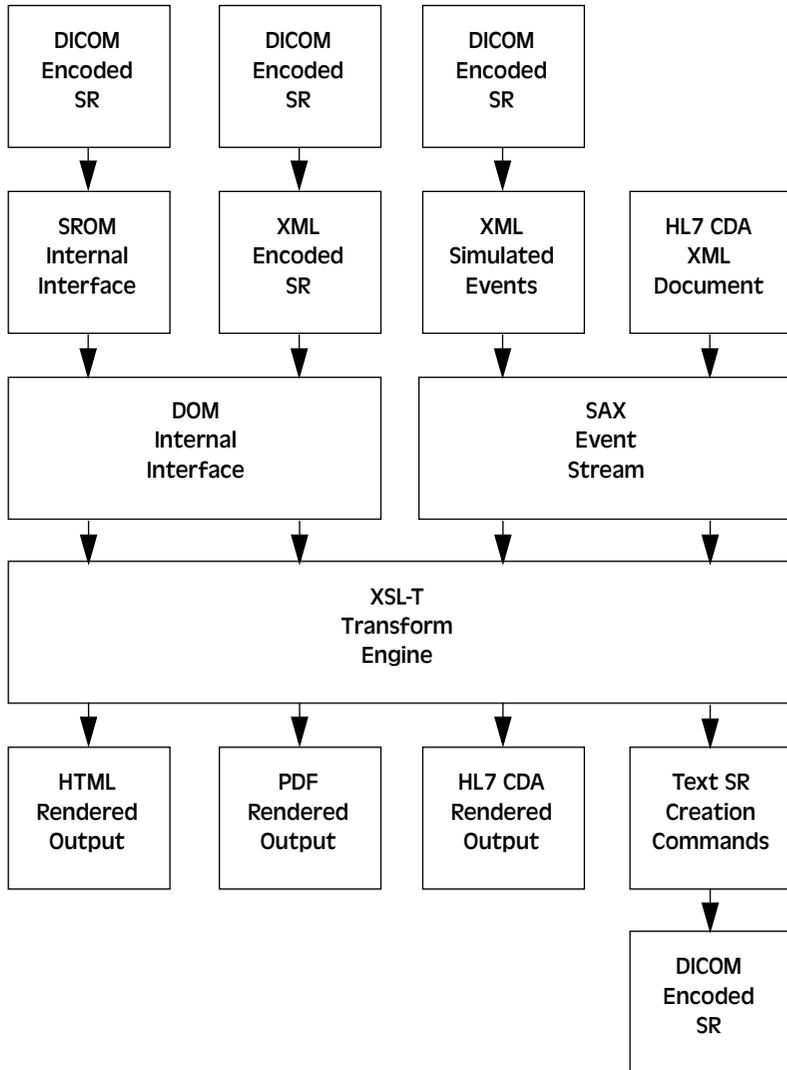


FIGURE 53. Potential Scenarios for Applying XSL-T

# ***Walking Through the Standard***

---

The DICOM standard currently consists of well over a thousand pages, organized in parts according to ISO Directives. Each part describes a single “layer,” each of which may be applicable to many services or objects. This “horizontal” organization is extremely intimidating for the newcomer or specialist who is interested only in a single “vertical” application.

Accordingly, this annex identifies those sections in the standard<sup>1</sup> that are of direct relevance to structured reporting, by purpose, part and section number.

Though this material will become dated as new SR IODs and SOP classes are added to the standard, the description of the locations of the generic information and the basic IODs and SOP classes will remain relevant.

## **Information Object Definitions**

Every DICOM object has an Information Object Definition (IOD) that describes what is contained within the object. These are defined in PS 3.3 aka. “part 3”.<sup>2</sup>

Annex A describes which modules are included in each IOD. In the case of structured reporting, the relevant sub-sections are in section A.35:

- ▶ A.35.1 Basic Text SR IOD
- ▶ A.35.2 Enhanced SR IOD
- ▶ A.35.3 Comprehensive SR IOD

Each of these sub-sections follows the same pattern, summarizing:

- 
1. *The 2000 edition of the standard is referenced, since it is the most recent, and it is the first to have Supplement 23, Structured Reporting, incorporated.*
  2. *PS 3 refers to the whole DICOM Standard, PS 3.x refers to part x of PS 3.*

- ▶ what modules are contained in the IOD (the same for all SR IODs)
- ▶ what Value Types are permitted in that IOD
- ▶ what Relationship Types are permitted in that IOD, specified per source and target content item value type
- ▶ any additional constraints

This is a very important section, since it is the *only* place in the standard where the different “specializations” of the generic SR framework are defined. The remainder of the description of SR in PS 3.3 is generic and applies to all three IODs.

Since SR objects are composite, the relevant modules are described in annex C of PS 3.3. Annex C occupies the bulk of PS 3.3, since it is where the real “meat” of each IOD is described in detail. The “general” modules that are included in all SR IODs (in addition to many other composite objects) are in section C.7 and C.12 and include:

- ▶ C.7.1.1 Patient Module
- ▶ C.7.1.2 Specimen Identification Module
- ▶ C.7.2.1 General Study Module
- ▶ C.7.2.2 Patient Study Module
- ▶ C.12.1 SOP Common Module

The modules that pertain only to SR are in section C.17 and include:

- ▶ C.17.1 SR Document Series Module
- ▶ C.17.2 SR Document General Module
- ▶ C.17.3 SR Document Content Module

In each module definition, attributes are named, identified by a group and element tag, described and specified by type. The meaning of the type (1, 2 or 3: whether an attribute is required, may be zero length, or is optional, respectively), is described in another part, PS 3.5, in section 7.4.

The macro used to define coded entries is described in Section 8, where the attributes in the macro are listed, and each attribute is described in detail. Annex D, which used to contain the list of coding schemes,<sup>3</sup> now contains an informative description of the principles behind coded entries and their use. This annex is based on one of Dean Bidgood’s articles.

Though no templates are yet defined in the standard, the macro and attributes used to record in a content item that a template has been instantiated are described in Section 9.

---

3. The list was “moved” from annex D into the SNOMED-DICOM Microglossary (SDM), context group 167. With the demise of the SDM the question arises as to where this list will be reincarnated and the answer is probably somewhere in PS 3.16, the proposed DICOM Terminology Mapping Resource that will be added in Supplement 53.

For internationalization and localization, the character repertoires that may be used in the Specific Character Set attribute of the SOP Common Module are listed in section C.12.1.1.2.

## **SOP Classes**

Every IOD needs to be matched with a Service Class to create an Service-Object Pair (SOP) Class. The Service Classes are defined in PS 3.4.

The Storage Service Class, which is used to transmit SR objects over the network, is described in PS 3.4 annex B. There is a list of all storage SOP classes in section B.5, which includes SOP classes corresponding to each of the IODs from PS 3.3:

- ▶ Basic Text SR SOP Class
- ▶ Enhanced SR SOP Class
- ▶ Comprehensive SR SOP Class

Since storage on media is defined as a separate SOP class in PS 3.4 annex I, there is a corresponding list in section I.4.

The Query-Retrieve Service Class is described in PS 3.4 annex C. It is a little different, in that it is not organized as a match between IODs from PS 3.3 and services. Rather it describes its own information model in terms of requests and responses using “identifiers” that are data sets in their own right.

How keys are defined and matched is described in section C.2. How identifiers are constructed and what behavior is expected of SCUs and SCPs is described in C.4.1 for C-FIND and C.4.2 for C-MOVE. The SOP classes that comprise the Study Root Query-Retrieve SOP Class group are defined in section C.6.2.1.

## **Data Dictionary**

PS 3.3 describes all the attributes that are used in SR objects in each of the modules. Which encoding to use for each attribute is described in PS 3.6, where all of the attributes (data elements) used in DICOM are listed in a huge table, sorted numerically.

This table specifies the Value Representation (VR) that is used to encode each attribute, i.e. whether it is some kind of string, UID, binary number or whatever.

It also specifies the Value Multiplicity (VM) for each attribute, that is how many values may be encoded in a single attribute instance. In the case of attributes in SR this is usually one, but there are exceptions. Note also that the VM specified in PS 3.6 may occasionally be overridden by an explicit description in PS 3.3 when an attribute is “specialized” where it is used in a module. Any VM specified in PS 3.3 always overrides the VM specified in PS 3.6.

## Encoding

PS 3.5 describes how attributes are organized and encoded into data sets to be transmitted on the network or stored in a file.

The different Value Representations for encoding each attribute are described in section 6.2. How the group and element numbers that define a data element, the Value Representation, the Value Length and the value itself are encoded is defined in section 7. Sequence encoding and nesting are of particular relevance to SR and are described in section 7.5.

The types (1, 2 and 3) used in PS 3.3 to indicate whether attributes in a module are required, may be zero length or are optional are described in section 7.4.

The use and construction of unique identifiers (UIDs) is described in section 9, annex B and annex C.

The transfer syntaxes that specify which encoding choices are selected and used are described in general terms in section 10 and then defined in detail in annex A. In the case of SR the compression transfer syntaxes for pixel data are not relevant. For SR, the likely choices will be the default implicit VR little endian transfer syntax described in sections 10.1 and A.1, usually used on the network, and the explicit VR little endian transfer syntax described in section A.2 that is used on media.<sup>4</sup>

Support for character repertoires used in string values is described in section 6.1. In that section, the interpretation of the values in Specific Character Set is described, and the use of code extension techniques (ISO 2022 escape sequences) and initial conditions are specified. The default character repertoire (US-ASCII or ISO 646) is actually listed in annex E. Notice that the allowed defined terms for Specific Character Set, and the character repertoires to which they correspond, are listed not in PS 3.5 but rather in PS 3.3 in the section on the SOP Common Module. This is because the attribute Specific Character Set is contained in the SOP Common Module, which in turn is contained in all composite objects. See PS 3.3 section C.12.1.1.2. Examples of the use of Japanese and Korean multi-byte character sets are described in PS 3.5 annexes H and I respectively.

## Files and Media

How DICOM data sets are encoded on media in files is described in PS 3.10, specifically in section 7. The File Meta Information header that is prepended to the data set is defined in section 7.1. To create compliant DICOM media, a directory of files is also required, the DICOMDIR. This is described conceptually in PS 3.10, but since it is a DICOM “object,” the corresponding IOD is defined in PS 3.3, in annex F. There is an example of a DICOMDIR in PS 3.10 annex A.

For media to be DICOM compliant, it must be created according to a Media Application Profile. These are all defined in PS 3.11. At the present time there is only one

---

4. CP 218 proposes to add a new compression transfer syntax that would apply to the entire data set and not just pixel data, and hence may be useful for structured reports.

standard application profile that can be used to record structured reports on media, and that is the General Purpose CD-R profile described in PS 3.11 annex D. In essence, that profile specifies that uncompressed data sets of any composite storage SOP class may be recorded on CD-R.

The specifications of physical media and logical file system are contained in PS 3.12. In the case of SR and the general purpose profile, the CD-R media is defined in annex F, where ISO 9660 Level 1 is specified for the file system. CD-Rs may be multi-session but not packet written.

## **Network**

Anyone who needs to transfer DICOM structured reports over the network will likely either be using utilities or tools from external sources, or will already know more about DICOM than this book can cover. Very likely most SR implementations will create DICOM “files” and hand them off to a tool to transmit them, or will have such files provided to them for display or rendering.

It suffices to say here that the DICOM messages that support the service classes are defined in PS 3.7. C-STORE is described in C.9.1.1, C-FIND in section C.9.1.2 and C-MOVE in section C.9.1.4. This part also describes association establishment in annex D. The upper layer services used to transmit messages over TCP-IP are described in PS 3.8. Just ignore all the ISO OSI stuff (which is never used) and skip directly to section 9. Notice that some of the message specific details of the user info items (needed to actually implement the upper layer protocol) are actually to be found in PS 3.7 D.3 rather than in section 9.3.3.3 or annex D of PS 3.8.

## **Conformance**

Any implementation claiming conformance to the standard must have a Conformance Statement. The format of such statements is described in PS 3.2. Additional SOP class specific requirements are listed with each service in PS 3.4.

Conformance statements will describe:

- ▶ which network SOP classes are supported and in which roles (SCU or SCP)
- ▶ which Media Application Profiles are supported and in which roles (FSC, FSU or FSR)

Remember that no other conformance to DICOM is possible. In particular, storing DICOM datasets in files and exchanging them outside the context of the DICOM network protocol or a media application profile does *not* conform to DICOM.



# ***Basic Encoding Principles***

---

It is beyond the scope of this book to describe every nuance of DICOM. However, it may be helpful to those interested in structured reporting but not yet familiar with DICOM to summarize the basic principles of encoding. This will allow the examples in the body of the book to make more sense.

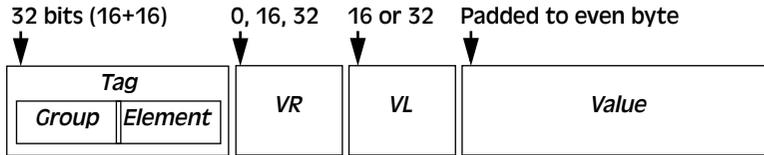
Interested readers should consult PS 3.5 of the standard itself. It is the one part of the standard that actually is quite straightforward to understand, even when read in isolation from the other parts.

## **Encoding**

DICOM objects are encoded in binary form as lists of attributes. Each attribute (or Data Element when one is speaking of encoding) is identified by a Tag consisting of a 16-bit group and 16-bit element number which together uniquely identify the data element. The data elements in the data set are sorted sequentially by ascending tag number, and each data element is sent only once. In other words, the data set is not encoded in the structure or order that is described in the IODs. When an IOD contains modules that define the same tag more than once (for example, to specialize its requirements or behavior), it is actually encoded only once. Those data elements in even numbered groups are standard DICOM elements. Those with odd group numbers are Private Data Elements, whose requirements and behavior are implementation-dependent.

The meaning of the value of a data element is defined by the IOD in which it is used. The form or Value Representation (VR) (i.e., whether it is a date or time or binary value or string or name, etc.) is defined by the Data Dictionary. The data dictionary is the part of the DICOM standard that defines the VR for all the standard data elements, as well as their Value Multiplicity (VM). The data dictionary itself is not transmitted with the data set. For example, one has to know from the standard that data

element (0010,0010) is Patient Name; there is nothing contained in the encoded data set to indicate this. Explicitly encoded in the data element is a field describing the length of the value in bytes called the Value Length (VL).<sup>1</sup> This is illustrated in Figure 54.



For example:

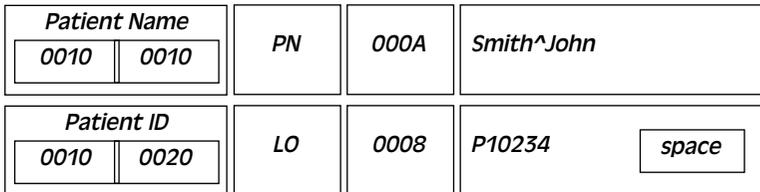


FIGURE 54. Data Element Encoding

The VR may or may not be explicitly encoded in the data set. In the ACR-NEMA standards, prior to DICOM, it was not: hence, the baseline or default encoding or transfer syntax is referred to as an Implicit VR transfer syntax. All network implementations of DICOM must support this default transfer syntax. All other transfer syntaxes explicitly encode a two-character field that identifies the VR of the data element. The available value representations are shown in the table that follows:

TABLE 20. Value Representations

AE	Application Entity	AS	Age String	AT	Attribute Tag
CS	Code String	DA	Date	DS	Decimal String
DT	DateTime	FL	Float Single	FD	Float Double
IS	Integer String	LO	Long String	LT	Long Text
OB	Other Byte	OW	Other Word	PN	Person Name

1. The number of bits used to encode the VR and VL fields depends on the transfer syntax and the VR. In the default implicit VR little endian transfer syntax, the VR is never sent and the VL is always 32 bits. In the all other transfer syntaxes, the VR is always sent and is 16 bits. For OB, OW, SQ, UT and UN VRs, the 16 bit VR is followed by 16 bits of zeroes, then a 32 bit VL. For all other VRs, a 16 bit VR field is followed by a 16 bit VL. The idea is to stay aligned on a 32 bit boundary when 32 bit VLs are required for “big” values, and save 32 bits for the others. In retrospect, this “optimization” has caused nothing but trouble, particularly when adding new VRs to the standard, or trying to send “big” values in “small” VRs.

TABLE 20. Value Representations

SH	Short String	SL	Signed Long	SQ	Sequence
SS	Signed Short	ST	Short Text	TM	Time
UI	Unique Identifier	UL	Unsigned Long	UN	Unknown
US	Unsigned Short	UT	Unlimited Text		

## Transfer Syntaxes

DICOM provides several different transfer syntaxes for transferring images across the network or for storing them in files. There are three broad classes of transfer syntaxes that include the:

- ▶ default Implicit Value Representation Little Endian Transfer Syntax
- ▶ Little and Big Endian Explicit Value Representation Transfer Syntaxes
- ▶ compression transfer syntaxes (which are all Explicit Value Representation)

The little and big endian choices are provided because some implementers believe there are performance advantages to using the native byte ordering of the host computers. In practice, few implementations support other than little endian. The currently available compression transfer syntaxes include lossless and lossy JPEG compression, lossless and near-lossless JPEG-LS, as well as a run length encoding (RLE) scheme popular with ultrasound vendors. None of the compression transfer syntaxes are of relevance to encoding SR documents themselves,<sup>2</sup> but may be encountered in images that are referenced by SR documents.

Which transfer syntax is used is either:

- ▶ negotiated when the association is established (in the case of network transfers)
- ▶ defined a priori in Media Application Profiles (in the case of storage on interchange media)

DICOM files stored on media always start with a File Meta Information header, which is always in Explicit VR Little Endian transfer syntax, and which specifies what transfer syntax has been used to encode the data set that follows. The meta information is never transmitted on the network.

2. CP 218 proposes to add the “deflate” compression scheme (used in the popular gzip and zip file formats) to compress the entire data set, particularly for structured reports.



## List of Codes

Table 21 lists all the codes used in the examples in this book. The coding scheme “99PMP” is a private scheme that was created for the purpose of this book. The coding scheme “DCM” is defined in DICOM itself. Codes from SNOMED are designated by “SNM3”, those from ICD9-CM, by “I9C”, those from ICD10-PCS by “I10P” and those from the Unified Code of Units of Measure by “UCUM”.

**TABLE 21.** List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
T-D4000	SNM3	Abdomen
209303	99PMP	Abdominal X-ray
209215	99PMP	Abnormal
209218	99PMP	Absent (not present)
209211	99PMP	Absent features
209701	99PMP	Admitted
209413	99PMP	Amorphous (calcification)
209104	99PMP	Anatomic Site (syn. anatomy)
R-10206	SNM3	AP
[arb'U]	UCUM	Arbitrary unit
209254	99PMP	Area
209802	99PMP	aV Leads
209049	99PMP	Baseline image
209050	99PMP	Best illustration of findings

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
209451	99PMP	Biparietal (diameter)
209400	99PMP	Bone lesion
209399	99PMP	Bones (NOS)
T-28800	SNM3	Both lungs
T-A0100	SNM3	Brain
T-04000	SNM3	Breast
209305	99PMP	C-spine X-ray
209225	99PMP	Calcifications
cm	UCUM	Centimeter
209236	99PMP	Central (location)
T-D3000	SNM3	Chest
T-D3000	SNM3	Chest
209302	99PMP	Chest X-ray
209076	99PMP	Chest X-ray Report
209300	99PMP	Chest X-ray,2V,PA,Lat
209077	99PMP	Chest X-ray,2V,PA,Lat Report
209210	99PMP	Clear (no findings)
209222	99PMP	Clustered
209409	99PMP	Codmans triangle
209408	99PMP	Complex
209005	99PMP	Conclusion
209006	99PMP	Conclusions
209032	99PMP	Coordinates
209796	99PMP	Coronary analysis result
209798	99PMP	Coronary stenosis
209651	99PMP	Country (of language)
209047	99PMP	Current (object or image)
209242	99PMP	Density (radiographic density of lesion)
209453	99PMP	Derivation
209011	99PMP	Description
209009	99PMP	Description of procedure
209018	99PMP	Diagnoses
209017	99PMP	Diagnosis
209019	99PMP	Diagnosis codes

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
209074	99PMP	Diagnostic Imaging Report
209230	99PMP	Diameter
209403	99PMP	Diaphyseal (location)
209026	99PMP	Differential diagnosis
209250	99PMP	Diffuse (distribution)
209223	99PMP	Distribution
209247	99PMP	Distribution
209909	99PMP	Document
209412	99PMP	Dystrophic (calcification)
209801	99PMP	End Systole
209241	99PMP	Endpoints
209239	99PMP	Epicenter
209401	99PMP	Epiphyseal (location)
209599	99PMP	Equivalent meaning of name
209600	99PMP	Equivalent meaning of value
F-10440	SNM3	Erect
209306	99PMP	Erect and supine abdominal X-ray
209301	99PMP	Erect, single view, PA chest X-ray (syn. Erect, PA chest X-ray)
T-56450	SNM3	Esophageal lumen
T-56000	SNM3	Esophagus
209022	99PMP	Exam
209212	99PMP	Existence negation flag
209304	99PMP	Facial bones X-ray
209243	99PMP	Fat (density)
209450	99PMP	Fetal head
209904	99PMP	Fetus
209902	99PMP	Fetus Number
209001	99PMP	Finding
209002	99PMP	Findings
209415	99PMP	Fluid level
209248	99PMP	Focal (distribution)
209702	99PMP	Functional condition
209601	99PMP	German meaning of value
209452	99PMP	Hadlock (technique to measure fetal biparietal diameter)

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
T-32000	SNM3	Heart
[hnsfU]	UCUM	Hounsfield unit
209023	99PMP	Image (of an exam)
209048	99PMP	Images
209046	99PMP	Images with findings
50291CC	I10P	IMAGING:CNS:CT:SELLA:LOWOSMOLAR:IT,U,E:2PLANE3D
209014	99PMP	Impression
209013	99PMP	Impressions
209024	99PMP	Indication
209003	99PMP	Initial findings
F-20010	SNM3	Inspiration
209139	99PMP	Intra-cavitary
209238	99PMP	Irregular (shape or margin)
209602	99PMP	Japanese meaning of value
209045	99PMP	Key image description
209406	99PMP	Lamellated
209650	99PMP	Language
209652	99PMP	Language of content item and descendants
209655	99PMP	Language of name
209653	99PMP	Language of name and value
209654	99PMP	Language of value
R-102CD	SNM3	Lateral
428.1	I9C	Left heart failure
T-32600	SNM3	Left ventricle
209245	99PMP	Lobular (shape)
209235	99PMP	Location
209258	99PMP	Lower limit (of normal range for measurement)
209251	99PMP	Lucent (density)
T-28000	SNM3	Lung
209800	99PMP	LV outline
209799	99PMP	LV outline at end systole
209228	99PMP	Malignancy
209226	99PMP	Malignant
209229	99PMP	Malignant mass

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
209237	99PMP	Margin
209227	99PMP	Mass
209455	99PMP	Mean
209402	99PMP	Metaphyseal (location)
209416	99PMP	Metastases
209108	99PMP	Method
209253	99PMP	Mixed (density)
209102	99PMP	Modality
209906	99PMP	Mother of Fetus
209079	99PMP	MRI Report
209249	99PMP	Multifocal (distribution)
209220	99PMP	Multiple
209246	99PMP	Multiplicity (of findings)
209231	99PMP	Normal
209214	99PMP	Normality (of finding)
209454	99PMP	Number of samples
209099	99PMP	Number of views
209031	99PMP	Nurse
209069	99PMP	Observer
209217	99PMP	Other (unspecified choice or finding)
209216	99PMP	Other features
209244	99PMP	Oval
109041	DCM	P wave
R-10214	SNM3	PA
209150	99PMP	PA and Lateral
209700	99PMP	Patient
209903	99PMP	Patient
%	UCUM	Percent
209797	99PMP	Percent stenosis
209029	99PMP	Performing physician
209404	99PMP	Periosteal reaction
209224	99PMP	Pleomorphic
209105	99PMP	Position (of subject wrt. gravity)
209213	99PMP	Presence (of property)

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
209219	99PMP	Present
209025	99PMP	Previous findings
209021	99PMP	Prior report
209007	99PMP	Procedure
209028	99PMP	Procedure date
209027	99PMP	Procedure type
209008	99PMP	Procedures
209140	99PMP	Projection (radiography)
209078	99PMP	Projection Radiography Report
209299	99PMP	Projection X-ray (syn. projection radiograph)
209907	99PMP	Quotation
209908	99PMP	Quotation Mode
209075	99PMP	Radiology Report
R-10218	SNM3	RAO (right anterior oblique projection)
209010	99PMP	Reason for exam
209016	99PMP	Recommendation
209015	99PMP	Recommendations
F-10450	SNM3	Recumbent
209103	99PMP	Region
209068	99PMP	Report
209073	99PMP	Report with images
209020	99PMP	Reported procedure
209233	99PMP	Round
209307	99PMP	Routine facial fracture screen
209252	99PMP	Sclerotic (density)
209309	99PMP	Screening mammogram
197.0	I9C	Secondary malignant neoplasm of lung
209044	99PMP	Seen on (as in "finding seen on image")
209240	99PMP	Selected volume
209232	99PMP	Shape
209221	99PMP	Single
209234	99PMP	Size
88.23	I9C	Skeletal x-ray of wrist and hand
209411	99PMP	Soft tissue calcification

TABLE 21. List of Codes Used

Code Value	Coding Scheme Designator	Code Meaning
209410	99PMP	Soft tissue mass
209405	99PMP	Solid
209106	99PMP	Special (property of X-ray examination)
209905	99PMP	Specimen
209107	99PMP	Structure
209702	99PMP	Study Instance UID
209900	99PMP	Subject Class
209901	99PMP	Subject Name
209414	99PMP	Subperiosteal resorption
209004	99PMP	Subsequent findings
209407	99PMP	Sunburst
209109	99PMP	Technique
209030	99PMP	Technologist
209256	99PMP	Time (point in time for measurement)
209098	99PMP	Trauma (for trauma, property of X-ray examination)
209308	99PMP	Trauma lateral c-spine X-ray
1	UCUM	Unity
209209	99PMP	Unknown
209257	99PMP	Upper limit (of normal range for measurement)
209910	99PMP	Verbal
209101	99PMP	View
209100	99PMP	Views
209255	99PMP	Volume



## ***List of Attributes***

---

The definitive reference for the data elements that correspond to the attribute used in the Structured Reporting IODs is DICOM PS 3.6. However, since this part of the standard contains all of the data elements for the entire standard, and is sorted only by tag number, it is a little hard to follow.

Accordingly, just the subset of data elements used in SR is included here. In addition to specifying the name, tag, VR and VM as in PS 3.6, the number of items permitted in each sequence as defined in PS 3.3 is also described.

The list is tabulated in two forms, one sorted by tag, and the other sorted by name.

## Attributes Sorted by Tag

TABLE 22. List of Attributes used in Structured Reporting, Sorted by Tag.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Study Date	(0008,0020)	DA	1	
Content Date	(0008,0023)	DA	1	
Study Time	(0008,0030)	TM	1	
Content Time	(0008,0033)	TM	1	
Accession Number	(0008,0050)	SH	1	
Retrieve AE Title	(0008,0054)	AE	1-n	
Modality	(0008,0060)	CS	1	
Manufacturer	(0008,0070)	LO	1	
Institution Name	(0008,0080)	LO	1	
Institution Address	(0008,0081)	ST	1	
Referring Physician's Name	(0008,0090)	PN	1	
Code Value	(0008,0100)	SH	1	
Coding Scheme Designator	(0008,0102)	SH	1	
Coding Scheme Version	(0008,0103)	SH	1	
Code Meaning	(0008,0104)	LO	1	
Mapping Resource	(0008,0105)	CS	1	
Context Group Version	(0008,0106)	DT	1	
Context Group Local Version	(0008,0107)	DT	1	
Code Set Extension Flag	(0008,010B)	CS	1	
Private Coding Scheme Creator UID	(0008,010C)	UI	1	
Code Set Extension Creator UID	(0008,010D)	UI	1	
Context Identifier	(0008,010F)	CS	1	
Station Name	(0008,1010)	SH	1	
Study Description	(0008,1030)	LO	1	
Procedure Code Sequence	(0008,1032)	SQ	1	1-n
Institutional Department Name	(0008,1040)	LO	1	
Physician(s) of Record	(0008,1048)	PN	1-n	
Name of Physician(s) Reading Study	(0008,1060)	PN	1-n	
Admitting Diagnoses Description	(0008,1080)	LO	1-n	
Manufacturer's Model Name	(0008,1090)	LO	1	
Referenced Study Sequence	(0008,1110)	SQ	1	0-n <sup>b</sup> or 0-1
Referenced Study Component Sequence	(0008,1111)	SQ	1	0-1

TABLE 22. List of Attributes used in Structured Reporting, Sorted by Tag.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Referenced Series Sequence	(0008,1115)	SQ	1	1-n
Referenced Patient Sequence	(0008,1120)	SQ	1	1
Referenced SOP Class UID	(0008,1150)	UI	1	
Referenced SOP Instance UID	(0008,1155)	UI	1	
Referenced Frame Number	(0008,1160)	IS	1-n	
Referenced SOP Sequence	(0008,1199)	SQ	1	1-n <sup>c</sup>
Patient's Name	(0010,0010)	PN	1	
Patient ID	(0010,0020)	LO	1	
Patient's Birth Date	(0010,0030)	DA	1	
Patient's Birth Time	(0010,0032)	TM	1	
Patient's Sex	(0010,0040)	CS	1	
Other Patient IDs	(0010,1000)	LO	1-n	
Other Patient Names	(0010,1001)	PN	1-n	
Patient's Age	(0010,1010)	AS	1	
Patient's Size	(0010,1020)	DS	1	
Patient's Weight	(0010,1030)	DS	1	
Ethnic Group	(0010,2160)	SH	1	
Occupation	(0010,2180)	SH	1	
Additional Patient's History	(0010,21B0)	LT	1	
Patient Comments	(0010,4000)	LT	1	
Device Serial Number	(0018,1000)	LO	1	
Software Versions	(0018,1020)	LO	1-n	
Spatial Resolution <sup>d</sup>	(0018,1050)	DS	1	
Date of Last Calibration	(0018,1200)	DA	1-n	
Time of Last Calibration	(0018,1201)	TM	1-n	
Study Instance UID	(0020,000D)	UI	1	
Study ID	(0020,0010)	SH	1	
Series Number	(0020,0011)	IS	1	
Instance Number	(0020,0013)	IS	1	
Pixel Padding Value <sup>e</sup>	(0028,0120)	US or SS	1	
Requested Procedure Description	(0032,1060)	LO	1	
Requested Procedure Code Sequence	(0032,1064)	SQ	1	0-1
Specimen Accession Number	(0040,050A)	LO	1	
Specimen Sequence	(0040,0550)	SQ	1	0-1
Specimen Identifier	(0040,0551)	LO	1	

TABLE 22. List of Attributes used in Structured Reporting, Sorted by Tag.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Specimen Type Code Sequence	(0040,059A)	SQ	1	0-1
Slide Identifier	(0040,06FA)	LO	1	
Measurement Units Code Sequence	(0040,08EA)	SQ	1	1
Requested Procedure ID	(0040,1001)	SH	1	
Placer Order Number/Imaging Service Request	(0040,2016)	LO	1	
Filler Order Number/Imaging Service Request	(0040,2017)	LO	1	
Relationship Type	(0040,A010)	CS	1	
Verifying Organization	(0040,A027)	LO	1	
Verification DateTime	(0040,A030)	DT	1	
Observation DateTime	(0040,A032)	DT	1	
Value Type	(0040,A040)	CS	1	
Concept Name Code Sequence	(0040,A043)	SQ	1	1
Continuity of Content	(0040,A050)	CS	1	
Verifying Observer Sequence	(0040,A073)	SQ	1	1-n
Verifying Observer Name	(0040,A075)	PN	1	
Verifying Observer Identification Code Sequence	(0040,A088)	SQ	1	0-1
Referenced Waveform Channels	(0040,A0B0)	US	2-2n	
DateTime	(0040,A120)	DT	1	
Date	(0040,A121)	DA	1	
Time	(0040,A122)	TM	1	
Person Name	(0040,A123)	PN	1	
UID	(0040,A124)	UI	1	
Temporal Range Type	(0040,A130)	CS	1	
Referenced Sample Positions	(0040,A132)	UL	1-n	
Referenced Time Offsets	(0040,A138)	DS	1-n	
Referenced Datetime	(0040,A13A)	DT	1-n	
Text Value	(0040,A160)	UT	1	
Concept Code Sequence	(0040,A168)	SQ	1	1
Measured Value Sequence	(0040,A300)	SQ	1	0-1
Numeric Value	(0040,A30A)	DS	1-n	
Predecessor Documents Sequence	(0040,A360)	SQ	1	1-n
Referenced Request Sequence	(0040,A370)	SQ	1	1-n
Performed Procedure Code Sequence	(0040,A372)	SQ	1	0-1
Current Requested Procedure Evidence Sequence	(0040,A375)	SQ	1	1-n
Pertinent Other Evidence Sequence	(0040,A385)	SQ	1	1-n

**TABLE 22.** List of Attributes used in Structured Reporting, Sorted by Tag.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Completion Flag	(0040,A491)	CS	1	
Completion Flag Description	(0040,A492)	LO	1	
Verification Flag	(0040,A493)	CS	1	
Content Template Sequence	(0040,A504)	SQ	1	1
Identical Documents Sequence	(0040,A525)	SQ	1	1-n
Content Sequence	(0040,A730)	SQ	1	1-n
Referenced Content Item Identifier	(0040,DB73)	UL	1-n	
Graphic Data	(0070,0022)	FL	2-n	
Graphic Type	(0070,0023)	CS	1	
Storage Media File-Set ID	(0088,0130)	SH	1	
Storage Media File-Set UID	(0088,0140)	UI	1	

- a. In DICOM PS 3.6, elements with a VR of SQ always have a VM of 1, regardless of how many items may be used in the sequence. The number of items that are permitted in a sequence is defined where it is used in PS 3.3.
- b. In General Study Module, 0-n; in SR Document General Module, 0-1.
- c. Except in Image Reference macro where only one presentation state may be referenced (optionally).
- d. This optional attribute is from a general module but is only applicable to images and not SR.
- e. This optional attribute is from a general module but is only applicable to images and not SR.

## Attributes Sorted by Name

**TABLE 23.** List of Attributes used in Structured Reporting, Sorted by Name.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Accession Number	(0008,0050)	SH	1	
Additional Patient's History	(0010,21B0)	LT	1	
Admitting Diagnoses Description	(0008,1080)	LO	1-n	
Code Meaning	(0008,0104)	LO	1	
Code Set Extension Creator UID	(0008,010D)	UI	1	
Code Set Extension Flag	(0008,010B)	CS	1	
Code Value	(0008,0100)	SH	1	
Coding Scheme Designator	(0008,0102)	SH	1	
Coding Scheme Version	(0008,0103)	SH	1	
Completion Flag	(0040,A491)	CS	1	
Completion Flag Description	(0040,A492)	LO	1	

TABLE 23. List of Attributes used in Structured Reporting, Sorted by Name.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Concept Code Sequence	(0040,A168)	SQ	1	1
Concept Name Code Sequence	(0040,A043)	SQ	1	1
Content Date	(0008,0023)	DA	1	
Content Sequence	(0040,A730)	SQ	1	1-n
Content Template Sequence	(0040,A504)	SQ	1	1
Content Time	(0008,0033)	TM	1	
Context Group Local Version	(0008,0107)	DT	1	
Context Group Version	(0008,0106)	DT	1	
Context Identifier	(0008,010F)	CS	1	
Continuity of Content	(0040,A050)	CS	1	
Current Requested Procedure Evidence Sequence	(0040,A375)	SQ	1	1-n
Date	(0040,A121)	DA	1	
Date of Last Calibration	(0018,1200)	DA	1-n	
DateTime	(0040,A120)	DT	1	
Device Serial Number	(0018,1000)	LO	1	
Ethnic Group	(0010,2160)	SH	1	
Filler Order Number/Imaging Service Request	(0040,2017)	LO	1	
Graphic Data	(0070,0022)	FL	2-n	
Graphic Type	(0070,0023)	CS	1	
Identical Documents Sequence	(0040,A525)	SQ	1	1-n
Instance Number	(0020,0013)	IS	1	
Institution Address	(0008,0081)	ST	1	
Institution Name	(0008,0080)	LO	1	
Institutional Department Name	(0008,1040)	LO	1	
Manufacturer	(0008,0070)	LO	1	
Manufacturer's Model Name	(0008,1090)	LO	1	
Mapping Resource	(0008,0105)	CS	1	
Measured Value Sequence	(0040,A300)	SQ	1	0-1
Measurement Units Code Sequence	(0040,08EA)	SQ	1	1
Modality	(0008,0060)	CS	1	
Name of Physician(s) Reading Study	(0008,1060)	PN	1-n	
Numeric Value	(0040,A30A)	DS	1-n	
Observation DateTime	(0040,A032)	DT	1	
Occupation	(0010,2180)	SH	1	
Other Patient IDs	(0010,1000)	LO	1-n	

TABLE 23. List of Attributes used in Structured Reporting, Sorted by Name.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Other Patient Names	(0010,1001)	PN	1-n	
Patient Comments	(0010,4000)	LT	1	
Patient ID	(0010,0020)	LO	1	
Patient's Age	(0010,1010)	AS	1	
Patient's Birth Date	(0010,0030)	DA	1	
Patient's Birth Time	(0010,0032)	TM	1	
Patient's Name	(0010,0010)	PN	1	
Patient's Sex	(0010,0040)	CS	1	
Patient's Size	(0010,1020)	DS	1	
Patient's Weight	(0010,1030)	DS	1	
Performed Procedure Code Sequence	(0040,A372)	SQ	1	0-1
Person Name	(0040,A123)	PN	1	
Pertinent Other Evidence Sequence	(0040,A385)	SQ	1	1-n
Physician(s) of Record	(0008,1048)	PN	1-n	
Pixel Padding Value <sup>b</sup>	(0028,0120)	US or SS	1	
Placer Order Number/Imaging Service Request	(0040,2016)	LO	1	
Predecessor Documents Sequence	(0040,A360)	SQ	1	1-n
Private Coding Scheme Creator UID	(0008,010C)	UI	1	
Procedure Code Sequence	(0008,1032)	SQ	1	1-n
Referenced Content Item Identifier	(0040,DB73)	UL	1-n	
Referenced Datetime	(0040,A13A)	DT	1-n	
Referenced Frame Number	(0008,1160)	IS	1-n	
Referenced Patient Sequence	(0008,1120)	SQ	1	1
Referenced Request Sequence	(0040,A370)	SQ	1	1-n
Referenced Sample Positions	(0040,A132)	UL	1-n	
Referenced Series Sequence	(0008,1115)	SQ	1	1-n
Referenced SOP Class UID	(0008,1150)	UI	1	
Referenced SOP Instance UID	(0008,1155)	UI	1	
Referenced SOP Sequence	(0008,1199)	SQ	1	1-n <sup>c</sup>
Referenced Study Component Sequence	(0008,1111)	SQ	1	0-1
Referenced Study Sequence	(0008,1110)	SQ	1	0-n <sup>d</sup> or 0-1
Referenced Time Offsets	(0040,A138)	DS	1-n	
Referenced Waveform Channels	(0040,A0B0)	US	2-2n	
Referring Physician's Name	(0008,0090)	PN	1	

TABLE 23. List of Attributes used in Structured Reporting, Sorted by Name.

Attribute Name	Tag	VR	VM	# of SQ Items <sup>a</sup>
Relationship Type	(0040,A010)	CS	1	
Requested Procedure Code Sequence	(0032,1064)	SQ	1	0-1
Requested Procedure Description	(0032,1060)	LO	1	
Requested Procedure ID	(0040,1001)	SH	1	
Retrieve AE Title	(0008,0054)	AE	1-n	
Series Number	(0020,0011)	IS	1	
Slide Identifier	(0040,06FA)	LO	1	
Software Versions	(0018,1020)	LO	1-n	
Spatial Resolution <sup>e</sup>	(0018,1050)	DS	1	
Specimen Accession Number	(0040,050A)	LO	1	
Specimen Identifier	(0040,0551)	LO	1	
Specimen Sequence	(0040,0550)	SQ	1	0-1
Specimen Type Code Sequence	(0040,059A)	SQ	1	0-1
Station Name	(0008,1010)	SH	1	
Storage Media File-Set ID	(0088,0130)	SH	1	
Storage Media File-Set UID	(0088,0140)	UI	1	
Study Date	(0008,0020)	DA	1	
Study Description	(0008,1030)	LO	1	
Study ID	(0020,0010)	SH	1	
Study Instance UID	(0020,000D)	UI	1	
Study Time	(0008,0030)	TM	1	
Temporal Range Type	(0040,A130)	CS	1	
Text Value	(0040,A160)	UT	1	
Time	(0040,A122)	TM	1	
Time of Last Calibration	(0018,1201)	TM	1-n	
UID	(0040,A124)	UI	1	
Value Type	(0040,A040)	CS	1	
Verification DateTime	(0040,A030)	DT	1	
Verification Flag	(0040,A493)	CS	1	
Verifying Observer Identification Code Sequence	(0040,A088)	SQ	1	0-1
Verifying Observer Name	(0040,A075)	PN	1	
Verifying Observer Sequence	(0040,A073)	SQ	1	1-n
Verifying Organization	(0040,A027)	LO	1	

- 
- a. In DICOM PS 3.6, elements with a VR of SQ always have a VM of 1, regardless of how many items may be used in the sequence. The number of items that are permitted in a sequence is defined where it is used in PS 3.3.
  - b. This optional attribute is from a general module but is only applicable to images and not SR.
  - c. Except in Image Reference macro where only one presentation state may be referenced (optionally).
  - d. In General Study Module, 0-n; in SR Document General Module, 0-1.
  - e. This optional attribute is from a general module but is only applicable to images and not SR.



# ***Structured Reporting Object Model***

---

This Annex contains a proposed interface definition for a Structure Reporting Object Model (SROM), as described in the chapter on “Trees, Traversal and Transformation.” It is based on the W3C DOM recommendation, level 1. Currently only a Java binding is defined.

## **Java**

```
public abstract class SROMException
    extends RuntimeException {
    public SROMException(short code,String message) {
        super(message);
        this.code=code;
    }
    protected short code;
    public short getCode() { return code; }
    public static final short InvalidParameter = 1;
}

public interface CodedEntry {
    public String getCodeValue();
    public String getCodingSchemeDesignator();
    public String getCodingSchemeVersion();
    public String getCodeMeaning();

    // compares code value, coding scheme designator only
    public boolean equals(Object obj);
}

public interface CodedEntryFactory {
    public CodedEntry createCodedEntry(
```

```
        String codeValue,  
        String codingSchemeDesignator,  
        String codingSchemeVersion,  
        String codeMeaning  
    );  
  
    public CodedEntry createCodedEntry(  
        String codeValue,  
        String codingSchemeDesignator,  
        String codeMeaning  
    );  
}  
  
public interface Node {  
    public CodedEntry getNodeName();  
    public String getNodeName();  
    public String getRelationshipType();  
  
    public Node getParentNode();  
    public Node getFirstChild();  
    public Node getFirstNamedChild(CodedEntry name);  
    public Node getLastChild();  
    public Node getPreviousSibling();  
    public Node getNextSibling();  
    public Node getNextNamedSibling(CodedEntry name);  
  
    public Node insertBefore(Node newnode, Node refnode)  
        throws SROMEException;  
    public Node replaceChild(Node newnode, Node oldnode)  
        throws SROMEException;  
    public Node removeChild(Node oldnode)  
        throws SROMEException;  
    public Node appendChild(Node newnode)  
        throws SROMEException;  
  
    public boolean hasChildNodes();  
  
    public Node cloneNode(boolean deep);  
}  
  
public interface ContainerNode extends Node {  
    public String getContinuityOfContent();  
}  
  
public interface CodeNode extends Node {  
    public CodedEntry getValue();  
}  
  
public interface NumericNode extends Node {  
    public String getValue();  
    public CodedEntry getUnits();  
}
```

```
}

public interface TextNode extends Node {
    public String getValue();
}

public interface PNameNode extends Node {
    public String getValue();
}

public interface DateNode extends Node {
    public String getValue();
}

public interface TimeNode extends Node {
    public String getValue();
}

public interface DateTimeNode extends Node {
    public String getValue();
}

public interface UIDRefNode extends Node {
    public String getValue();
}

public interface CompositeNode extends Node {
    public String getSOPClassUID();
    public String getSOPInstanceUID();
}

public interface ImageNode extends CompositeNode {
    public int[] getFrameNumbers();
    public String getPresentationSOPClassUID();
    public String getPresentationSOPInstanceUID();
}

public interface WaveformNode extends CompositeNode {
    public int[] getChannelNumbers();
}

public interface SCoordNode extends Node {
    public String getGraphicType();
    public double[] getGraphicData();
}

public interface TCoordNode extends Node {
    public String getRangeType();
    public int[] getSamplePositions();
    public double[] getTimeOffsets();
    public String[] getDateTimes();
}
```

```
}

public interface ReferenceNode extends Node {
    public Node getNode();
}

public interface NodeFactory {
    public ContainerNode createContainerNode(
        CodedEntry name,
        String relationshipType,
        String continuityOfContent);

    public TextNode createTextNode(
        CodedEntry name,
        String relationshipType,
        String value);

    public CodeNode createCodeNode(
        CodedEntry name,
        String relationshipType,
        CodedEntry value);

    public NumericNode createNumericNode(
        CodedEntry name,
        String relationshipType,
        String value,
        CodedEntry units);

    public PNameNode createPNameNode(
        CodedEntry name,
        String relationshipType,
        String value);

    public DateNode createDateNode(
        CodedEntry name,
        String relationshipType,
        String value);

    public TimeNode createTimeNode(
        CodedEntry name,
        String relationshipType,
        String value);

    public DateTimeNode createDateTimeNode(
        CodedEntry name,
        String relationshipType,
        String value);

    public UIDRefNode createUIDRefNode(
        CodedEntry name,
        String relationshipType,
```

```
        String value);

    public CompositeNode createCompositeNode(
        CodedEntry name,
        String relationshipType,
        String SOPClassUID,
        String SOPInstanceUID);

    public ImageNode createImageNode(
        CodedEntry name,
        String relationshipType,
        String SOPClassUID,
        String SOPInstanceUID,
        int[] frameNumbers,
        String presentationSOPClassUID,
        String presentationSOPInstanceUID);

    public WaveformNode createWaveformNode(
        CodedEntry name,
        String relationshipType,
        String SOPClassUID,
        String SOPInstanceUID,
        int[] channelNumbers);

    public SCoordNode createSCoordNode(
        CodedEntry name,
        String relationshipType,
        String graphicType,
        double[] graphicData);

    public TCoordNode createTCoordNode(
        CodedEntry name,
        String relationshipType,
        String rangeType,
        int[] samplePositions,
        double[] timeOffsets,
        String[] dateTimes);

    public ReferenceNode createReferenceNode(
        /* note that there is no name */
        String relationshipType,
        Node node) throws SRMException;
}
```



---

## *Index*

---

### **A**

- Acquisition context 126, 131, 149, 156–158, 171, 175
  - encoding in SR 158–159, 171
- ANSI X3.50 50
- ASCII 73, 86, 251–252, 254, 256–258, 260, 262–264, 275, 297–299, 301, 306, 311, 314, 339, 348
- ASN.1 (Abstract Syntax Notation 1) 108, 286, 306
- Audio file formats
  - AIFF 32, 71
  - WAV 32, 71

### **C**

- Cascading Style Sheet (CSS) 31, 339
- CEN TC 251 237
- COAS (Clinical Observations Access Service) 296, 320–323
- Code meaning 35, 40–42, 52–53, 58–60, 62, 71, 107, 131, 134–135, 157–158, 176, 204–205, 215, 234, 236, 245, 251, 264–267, 272, 274, 308, 316, 336, 364, 367
- Code sequence macro 41–42, 55, 60–61, 75, 247, 333
- Code value 35, 39–42, 45, 50, 53–54, 58–60, 62, 71, 132, 134–135, 215, 234, 267, 274, 336–337, 364, 367
- Coding scheme designator 35, 39–43, 53, 58–60, 62, 71, 134, 234, 267, 274, 336–337, 364, 367
  - 99DEV 42
  - 99PMP 49
  - 99SDM 39, 42
  - HL7 316
  - implicit version 60
  - private 42, 61, 63
  - SNM3 40, 42, 44, 62
  - SRT 46
  - value representation (16 character limit) 45
- Coding scheme version 41–42, 60, 364, 367

---

Conformance 31–32, 54, 63, 71, 139, 161–162, 164–165, 167–169, 197, 213, 224, 226, 232, 264, 287, 296, 301, 303, 321, 349

Content Items 32–34, 65

- acquisition context 159
- and digital signatures 287
- and SR Object Model 333
- and templates 233
- and XML transcoding 310
- coded entry. See Value Type
- composite object reference. See Value Type
- concept modifiers 35, 114, 267
- concept modifiers and alternative meanings 273
- concept modifiers and language 271
- concept names 50
- containment propagating headings 115
- date, time, date-time. See Value Type
- Document Content Macro 109
- flat list 33, 105–106, 233
- identification 129
- image object reference. See Value Type
- image references 139
- implicit numbering 108
- name-value pairs 65, 71
- nodes of a tree 107
- numeric without value 76
- numeric. See Value Type
- observation context 176, 178, 182
- observation context and inheritance 177
- Observation DateTime 185
- person name. See Value Type
- properties conveyed by relationships 34
- purpose of reference. See Purpose of Reference
- recursive nesting 117
- references to other content items 33, 126–127
- references within same document only 129
- relationship required for all except root 112
- relationship type restriction by IOD 346
- relationships to other content items. See Relationships.
- root. See Root content item
- simple data types 102
- source of relationship 110, 123, 127
- spatial coordinates. See Value Type
- specification of by-reference relationship 127
- specification of relationship 127
- specifying relationship 112
- target of relationship 110, 123, 127
- template identification 346
- temporal coordinates. See Value Type
- text. See Value Type
- UID reference. See Value Type
- values 65

---

waveform object reference. See Value Type  
Coordinates 32–34, 65, 70, 91–94, 97–98, 100–102, 114, 122–123, 126, 130, 139, 144–147,  
150–151, 163–164, 238, 242–245, 317, 320

## D

Digital signatures 196, 228, 278–279, 283–289, 291–292  
Document imaging 152  
Document Object Model (DOM) 321, 325, 328–329, 331–332, 334–337, 339, 341–342,  
344, 373  
Document title 33, 35, 65–67, 69, 164, 178, 215  
DTD (Data Type Definition). See XML (eXtensible Markup Language)

## E

ECMA 252, 254  
External object references 32, 71, 168  
internal nodes 33, 129

## F

File Meta Information 299, 301, 305, 348, 353  
File Transfer Protocol (FTP) 322

## G

Graphic type 92–94, 97, 367–368  
CIRCLE 93, 95, 145, 244–245  
ELLIPSE 93, 95, 244–245  
MULTIPOINT 92–93, 95, 124, 244  
POINT 93, 95, 123, 144  
POLYLINE 92–93, 95–96, 244–245

## H

HIPAA (Health Insurance Portability and Accountability Act) 291–292  
HL7 41–42, 44–46, 59, 78, 100, 148, 152, 189, 202, 204, 209, 224, 226, 236–237, 295–301,  
304–306, 312–316, 321–323, 339, 344  
ED (Encapsulated Data) type 298–301, 314  
OBR segments 304–305, 321  
OBX segments 209, 298–300, 303–305, 313, 321  
ORU messages 209, 298, 305  
HyperText Markup Language (HTML) 31, 33, 71, 138, 229, 307, 312–313, 315, 320, 339–  
340, 342, 344  
HyperText Transfer Protocol (HTTP) 138, 299, 303, 322

## I

IHE (Integrating the Health-care Enterprise) 36, 101, 140, 175, 195, 224–226  
Image file formats  
GIF 32, 71, 138  
TIFF 32, 71, 152–153  
Information Object Definition. See IODs  
IOD 164, 187, 345–346  
and Acquisition Context 158  
and data elements 351  
and data set encoding 351  
and default observation context 175, 179–180

---

and Media Storage SOP Classes 348  
and modality 161  
and Modules 346  
and pixel data 162  
and Relationship Types 346  
and SOP Classes 112, 161–162, 187, 347  
and sources of context 58  
and Storage SOP Classes 161  
and Templates 31, 58  
and value sets 58  
and Value Types 346  
Basic Text and Enhanced SR 116  
Basic Text SR 345  
Composite 31–32, 70, 161, 232  
Comprehensive SR 345  
Conformance 165  
CT Image 161  
Enhanced SR 164, 345  
Mammography Image 232  
MR Image 161  
Structure Reporting IODs 31  
Structured Reporting 32, 109, 115, 163–164, 306, 345–346, 363  
Waveform 148

IODs 161  
  Basic Text SR 163  
  Composite 161  
  Comprehensive SR 163  
  Enhanced SR 163

ISO 2022 53, 73, 80, 252–253, 257–259, 261, 266, 348  
ISO 2955 50  
ISO 3166 269  
ISO 639 269  
ISO 646 86, 252, 254, 256–260, 348  
ISO 8859 73, 252–254, 256–257, 259, 263, 265, 311, 314  
ISO 9660 349  
ISO 9834 188  
ISO OSI 162  
ISO TC 215 237  
ISO+ 50

**J**  
JBIG 153  
JIS X 0201 256  
JIS X 201 256–258, 261  
JPEG 138, 153, 353  
JPEG-LS 153, 353

**K**  
Key image 30, 36, 101, 140–141, 225

---

## M

### Macro

Document Content 109, 112, 115

Document Relationship 109–110

SOP Instance Reference 200

Media Application Profile 162, 303, 348–349, 353

MIME (Multipurpose Internet Mail Extensions) 299–301

and COAS 322

and HL7 CDA 313

and HL7 separators 301

and http 303

Base64 encoding 298–301, 314

content type 299

content type for DICOM 299, 301–303

in HL7 ED 300

multipart 300

new line characters 301

### Module

Acquisition Context 75

SR Document Content 109, 193, 313, 346

SR Document General 36, 158–159, 172, 192–193, 198–200, 202–203, 207–208, 215, 287, 346, 367, 371

SR Document Series 204, 207, 346

## N

### Natural language

ambiguity 39

parsing 22, 114, 150, 330, 339

rendering 59, 166, 237

## O

### Observation context

observer 171–172, 178, 181, 183–184

procedure 171, 174, 178, 183–184

subject 171, 173, 178, 183–184

OCR (Optical Character Recognition) 152, 155

### Order

and modifiers 114

of content items 34, 108

of content items and presentation 108

of content items and references 129

of data elements 77, 87, 351

of data elements and digital signatures 288–289

of headings 68, 108, 238

of person name components 78

of sequence items 108

of words in code meaning 53

## P

Photometric interpretation 153

Presentation states 30, 88–89, 92, 94–95, 139–145, 154, 162, 168, 218, 224, 367, 371

---

## Q

- Query-Retrieve 37, 83, 200, 209, 347
  - and international character sets 265
  - hierarchical and relational models 200, 210–211
  - key matching 79, 86, 106, 209–212, 214–215, 265
  - query keys 214
  - query level 212
  - sequence matching 215
- Quoted documents 33, 171, 182–184, 193
- Quoted speech 182

## R

- Relationship Type 110–114, 127, 159, 164, 175, 346
  - and query matching 216
  - and SROM 334
  - CONTAINS 112–116, 120–123, 129, 134, 159, 177
  - encoded with child 117
  - HAS ACQ CONTEXT 113, 131, 158
  - HAS CONCEPT MOD 35, 53–54, 66, 113, 115, 131–132, 134–135, 215, 265, 267, 270–273
  - HAS OBS CONTEXT 113, 115, 130–131, 175, 178, 240
  - HAS PROPERTIES 110, 113–114, 121–123, 128, 134, 159, 163, 177, 240, 269–270, 272, 317
  - INFERRED FROM 34, 110, 113–114, 122–124, 126, 128, 159, 163, 177, 180, 239
  - SELECTED FROM 84, 92–93, 98, 102, 113–114, 126, 130, 139, 144–145, 164
  - value representation 113, 175
- Relationships 29, 31, 34, 105, 110, 114
  - and SR Object Model 326–327, 331–332
  - and templates 231, 233–234
  - and unambiguous rendering 165
  - by reference 110, 126–127, 159
    - encoding 127
    - forbidden for observation context 177
    - referenced content item identifiers 129
    - SOP classes 159, 163
  - by value 110, 127
  - by value or reference 34
  - choice of type 114
  - encoded with child 110
  - for structural purposes 114
  - in diagrams 111
  - in other standards 34, 310, 313, 315–317, 322–323
  - parent content item 110
  - rationale for choice of types 113
  - requirements 110
  - SOP class restrictions 165
  - source content item 110
  - target content item 110
  - types 34
- RLE 153, 353

---

Root content item 33, 35, 65, 67, 106–110, 112, 116–117, 129, 159, 168, 175, 178–179, 184, 215–217, 240, 270–271, 287, 326–327, 333–334, 336

Root node. See Root content item

## S

### Security

access control 278–279, 281–283, 291  
authentication 226, 279, 281, 285, 289  
confidentiality 226, 278–279, 283, 289–291, 313  
digital signatures. See Digital signatures  
integrity 278–279, 284  
MAC (Message Authentication Code) 283, 287–289  
MD-5 287  
RIPEMD-160 287  
RSA 287  
SHA (Secure Hash Algorithm) 287  
X.509 Certificates 286, 288, 306

### Service Classes

Storage 31, 161

Service-Object Pair (SOP) 31, 161, 187

Simple API for XML (SAX) 325, 328–329, 338–339, 342, 344

SOP Class 31–32, 67, 70, 74–75, 77–78, 81, 85, 87, 89, 91, 112, 122, 125, 134, 147–148, 159, 161–162, 164–165, 168, 187–188, 210, 272, 299

12-Lead ECG Waveform Storage 90, 149

and Language 270

and Templates 231–232, 247

Basic and Enhanced SR Storage SOP Classes 110

Basic Text and Enhanced SR Storage SOP Classes 33, 67, 116, 125, 134, 270

Basic Text SR Storage 87, 122, 163, 226, 347

Basic Voice Audio Waveform Storage 32, 71, 149, 151

C-Find SOP Classes 209

C-Get SOP Classes 210

C-Move SOP Classes 209

Composite 36, 349

Comprehensive SR Storage 121, 125, 163, 181, 226, 270, 347

Conformance 224, 226, 264, 301, 349

CT Image Storage 32, 71, 161

Detached Study Management 84

Enhanced SR Storage 163, 226, 347

For workflow management 220, 222

Future Secondary Capture Image Storage SOP Classes 153

Future SOP Classes 169

Future SR Storage SOP Classes 154, 163, 345

Grayscale Softcopy Presentation State Storage 162

Image Storage SOP Classes 153, 162, 164–165

Interpretation Worklist SOP Classes 222

Mammography CAD Storage 31, 163, 231, 247

Media Storage SOP Classes 162, 347

Modality Performed Procedure Step 38, 42, 221

Modality Worklist 38, 42, 221, 265

MR Image Storage 161

---

Multi-frame image storage SOP Classes 146  
Private SOP Classes 169  
Query-Retrieve SOP Classes 209, 347  
Secondary Capture Image Storage 152  
SR Storage SOP Classes 31, 36, 70, 112, 115, 130, 162–165, 231, 247, 306  
Standalone Curve Storage 162  
Standalone Overlay Storage 153, 162  
Standard Extended SOP Classes 213  
Storage SOP Classes 161–162, 168–169, 347, 349  
Support for referenced SOP Classes 168–169  
Support for retrieved SOP Classes 210  
Waveform Storage SOP Classes 148, 162, 210  
XA Image Storage 89  
SOP Class UID. See Unique Identifiers (UIDs)  
SOP Instance UID. See Unique Identifiers (UIDs)  
SOP. See Service-Object Pair (SOP)  
Speech recognition 36, 149–151, 339  
SSL (Security Sockets Layer) 279, 286  
Standard Generalized Markup Language (SGML) 307, 340  
Storage Service Class. See Service Classes  
Structured Reporting Object Model (SROM) 325, 334, 337, 339, 344, 373

## T

T.4 153

T.6 153

Templates 31–32, 55, 58, 70, 97, 103, 147, 167, 231, 235, 237–238, 241, 247–248, 306, 322  
    analogy with Modules 232  
    ancestors 235  
    and conformance 231–232  
    and presentation 232, 237  
    and relationship types 233–234  
    and rendering 167  
    and SOP Classes 163, 231–232, 247  
    and style sheets 237, 248  
    and units of measurement 244  
    and validation 236  
    complexity 236  
    conditions 234  
    constraining units of measurement 240  
    defined outside DICOM 237  
    definition 231  
    description of attributes 236  
    description of bone lesion 241–242  
    description of code meaning 236  
    description of mass 240  
    example 233–235  
    features 235  
    footnotes 236  
    for concept modifiers and meaning 267  
    for encoding tables 314  
    for measurements 240, 242–243

---

---

for measurements and temporal coordinates 244  
for measurements of area 244–245  
for measurements of diameter 244  
for measurements of distance 244  
for measurements of volume 244  
for negation 166, 246  
for normal range of measurements 322  
for Observation Context 171, 176  
for Procedure Context 174  
formal syntax 236  
identification 232, 235, 247–248, 346  
identification and nested templates 248  
identification and style sheets 248  
identification of subordinate templates 247  
include directive 234  
inclusion 234, 242  
inheritance 242  
invocation 233–234  
layout 231, 233–234, 241  
life without templates 237  
Mammography CAD 244  
multiple nesting levels 235  
patterns for IHE Year 2 225  
recursion 235  
single level 234  
sources of templates 236  
specialization 241, 243  
Supplement 53 231  
tail-end iteration 236  
Template Identification Macro 248, 346  
useful templates 238

Temporal range type 97–98, 366, 370  
  BEGIN 98  
  END 98  
  MULTIPOINT 98  
  MULTISEGMENT 98  
  POINT 98, 146–147  
  SEGMENT 98

Text  
  formatted 72, 304, 313  
  plain 29, 32, 34–36, 39, 52, 54, 65, 67–68, 72, 74, 108, 114, 116, 122, 139, 150, 239, 264,  
    273, 298, 303–307, 313–314, 316, 321–322, 339  
  pre-formatted 33, 71, 320

TIS 620 254, 263  
TLS (Transport Layer Security) 279–286, 289

**U**  
UID. See Unique Identifiers (UIDs)  
Unicode 263–264, 311, 314  
Uniform Resource Locator (URL) 322  
Unique Identifiers (UIDs) 36, 85–86, 161, 187–188, 210, 214, 347

---

root 188–189  
SOP Class UID 86, 88–90, 188, 200, 287  
SOP Instance UID 84–86, 88–90, 94, 130, 153, 169, 188–189, 191–194, 199–200, 207,  
212–213, 287, 305

## V

### Value Type 346

CODE 32, 58, 70–71, 74–75, 80, 86, 116, 128, 159, 164, 167, 217, 246, 270, 274, 333  
COMPOSITE 70, 83–84, 86–88, 90, 100–101, 130, 183, 189, 199, 333  
CONTAINER 65–72, 74, 106–107, 114–117, 120–121, 123, 126, 129, 134, 139–141, 159,  
163–164, 166–167, 175–177, 182, 216, 239, 241, 269, 303, 305, 314, 316, 333, 337, 342  
DATE 70, 80–81, 333  
DATETIME 70, 80–81, 98, 333  
IMAGE 70, 83–84, 87–90, 92–93, 98–102, 139–140, 144, 189, 199, 333  
NUM 32, 65, 70, 76–77, 116, 164, 226, 237, 239, 243, 318, 322, 333  
PNAME 32, 65, 70–71, 77, 80, 175–176, 308, 330–331, 333  
SCOORD 70, 84, 92–93, 95, 98, 100–102, 139–140, 144, 333  
TCOORD 70, 84, 90, 97–101, 139–140, 145, 333  
TEXT 32, 65, 70, 73–74, 116, 121–122, 140–141, 164, 166–167, 217, 225, 239, 244, 270,  
273–274, 322, 333, 337  
TIME 70, 80–81, 333  
UIDREF 70, 83–86, 88, 90–91, 100–101, 104, 333  
WAVEFORM 70, 83–84, 87–88, 90–92, 98, 100–101, 140, 145, 148, 189, 199, 333

## W

Wax pencil 137, 144, 153

## X

XML (eXtensible Markup Language) 31, 34, 110, 121, 218, 229, 296, 298, 306–314, 325,  
328–331, 337–340, 342, 344  
and binary data 311  
and browsers 138, 229  
and CORBAMed Clinical Observations Access Service (COAS) 321  
and digital signatures 313  
and HL7 Clinical Document Architecture (CDA) 152, 306, 312–313, 315, 339  
and Java 313, 328, 333, 341  
and Simple API for XML. See Simple API for XML (SAX)  
and style sheets 312  
and Unicode 264, 311  
character encoding 311  
character sets 314  
document 31, 312, 314, 321, 328–329, 338–339  
Document Object Model. See Document Object Model (DOM)  
Document Type Definition (DTD) 31, 306, 332  
documents 307, 312  
elements 308–309, 312, 315, 342  
elements vs. attributes 310  
encoding coded entries 315–316  
encoding SR 308, 310–312, 315, 329–330, 340, 342  
eXtensible Style Sheet - Formatting Objects (XSL-FO) 312, 340  
eXtensible Style Sheet - Transformations (XSL-T) 229, 237, 312, 340–342, 344  
eXtensible Style Sheet (XSL) 31, 229, 312, 339–340, 342

---

namespaces 309  
parsed character data (PCDATA) 311  
parsers 264, 311, 314, 342  
pointers 310  
Schema 31, 306, 309–312  
tools 312  
W3C Recommendation 31  
well-formed document 31, 311  
XLink 310  
XPath 218, 310, 312, 340  
XPointer 310  
XSL. See XML (eXtensible Markup Language)



---

## *About the Author*

---

David Clunie is currently the industry co-chairman of the DICOM Committee, editor of the DICOM standard, as well as a member or chairman of several of the DICOM working groups, including digital x-ray, compression, interchange media, base standard, display, mammography and clinical trials. He has recently joined ComView Corporation as Director of Development, Medical Imaging Products. Formerly he was Director of Medical Imaging Technology at Quintiles Intelligent Imaging and before that, Lead Designer for DICOM Standardization at GE Medical Systems.

Born in Scotland but raised and educated in Australia, in a previous life he was a neuro-radiologist, but decided to forsake the rewards of medical practice for the excitement of developing communications technology standards. Having travelled and worked in Australia, the Middle East and the US, he and his wife Eleanor now live in rural western Pennsylvania, both telecommuting as much as possible, and occasionally venturing into New York City in search of excitement or supplies.



---

## *Colophon*

---

This book was written in Adobe Framemaker on various Apple Macintosh systems, the author having recently foresworn the use of more popular but less reliable word processing applications as the result of previous nasty experiences.

Despite the author's British origin, the American spelling of words like gray and color are used throughout. This is not intended as a betrayal of heritage, but rather is to be consistent with the spelling used in the DICOM standard itself.

Most of the text is set in Chantilly. Antique Olive is used for headings and in illustrations. The OcrB font is used for examples of pseudo-code and reports. Japanese characters are Kozuka Minchu. Korean Hangul characters are Munwha Ba.

The manuscript was first written to Postscript and converted to PDF using Adobe Distiller before being sent to the printer.

No animals were harmed during the production of this book, and no Microsoft products were used.

