DICOM

**Digital Imaging and Communications in Medicine** (**DICOM**) is a standard for storing and [transmitting](https://en.wikipedia.org/wiki/Data_transmission) [medical images](https://en.wikipedia.org/wiki/Medical_imaging) enabling the integration of medical imaging devices such as scanners, servers, workstations, printers, network hardware, and [picture archiving and communication systems](https://en.wikipedia.org/wiki/Picture_archiving_and_communication_system) (PACS) from multiple manufacturers. It has been widely adopted by [hospitals](https://en.wikipedia.org/wiki/Hospital).

DICOM devices:

A medical device supporting and implementing the DICOM standard is defined as a **DICOM-compliant device**. A “DICOM-compliant” device may be an acquisition device (e.g., CR equipment, CT equipment, MR equipment, etc.), or a workstation, or a server, or any other kind of device able to connect to the DICOM network exchange data with other nodes using the DICOM protocol. For this reason, almost all DICOM devices need to have a network interface.

DICOM devices attached to a DICOM network are often referred to also as **DICOM nodes** or **DICOM peers**.

PACS:

**PACS** stands for **P**icture **A**rchiving and **C**ommunication **S**ystem. It can generally be defined as the whole system managing medical images and related data in a DICOM-compliant way.

Although there are different interpretations about which components fall within the definition of “PACS” (some of them are more comprehensive, some others are more restrictive), here is a list of entities which are normally considered to belong to a PACS:

* The acquisition equipment (also known as **modalities**)
* The images archive (also known as **PACS server**)
* The diagnostic workstations
* The network connecting the above components

All the above components must be DICOM-compliant devices, in order to be able to cooperate and exchange data also in a cross-vendor scenario.

[**NeoLogica’s LogiPACS software**](https://www.neologica.it/eng/Products/LogiPACS) is a notable example of a full-featured PACS server software, supporting archiving of millions of DICOM images as well as many other advanced features.

DICOM service:

The DICOM standard specifies several images-related services which are useful in the medical imaging workflow.  
Here is a short list of the most frequently-used DICOM services:

* [Verification service](https://www.neologica.it/eng/Tutorial/DICOMVerification)
* [Storage service](https://www.neologica.it/eng/Tutorial/DICOMStorage)
* [Storage Commitment service](https://www.neologica.it/eng/Tutorial/DICOMStorageCommitment)
* [Query/Retrieve service](https://www.neologica.it/eng/Tutorial/DICOMQueryRetrieve)
* Print service
* Modality Worklist service
* Modality Performed Procedure Step service

The DICOM standard describes and defines these services both at the *semantic* level and at the *protocol* level.

Each DICOM service implies the communication between the **Service Class Provider** (SCP)and **Service Class User (SCU)** entities

DICOM Verification:

The DICOM Verification service is probably the simplest DICOM service. It is used to verify DICOM connectivity between two DICOM nodes. Basically, it is the DICOM-equivalent of the “ping” network command, and indeed it is often referred to as **DICOM ping**.

The DICOM Verification service follows the usual SCP / SCU pattern: a DICOM Verification SCP acts as a server and waits for incoming connections by DICOM Verification SCUs. In turn, a DICOM Verification SCU, in order to verify the DICOM connectivity with another node acting as a DICOM Verification SCP, actively connects to that SCP node, sends the verification request, and waits for the verification response by the SCP.

C-ECHO-RQ (request), C-ECHO-RSP (response).

The DICOM Verification service is often used during the initial configuration phase of new DICOM nodes, in order to ensure correct communication with the other nodes of the DICOM network.

How to read DICOMs using ImageJ:

As ImageIO and JAI do not have support form DICOMs, you will need a third party alternative. For example ImageJ ([download](https://imagej.nih.gov/ij/download.html) and [Javadoc](https://imagej.nih.gov/ij/developer/api/)) provides a DICOM decoder and viewer. The application provides a user interface which can be used to decode and view DICOM files. Alternatively by adding the JAR file (included in the download) to your Java project you can use the [DICOM](https://imagej.nih.gov/ij/developer/api/) class to decode and open DICOM files

DICOM dcm = **new** DICOM();

dcm.open("image.dicom");

**if** (dcm.getWidth()==0)

IJ.log("Error opening image.dicom");

**else**

dcm.show();

**DicomParser:**

dicomParser is a lightweight library for parsing DICOM P10 byte streams in modern HTML5 based web browsers (IE10+), Node.js and Meteor. dicomParser is fast, easy to use and has no required external dependencies.

**packaged source file:**

* [dicomParser.js](https://raw.githubusercontent.com/chafey/dicomParser/master/dist/dicomParser.js)
* [dicomParser.min.js](https://raw.githubusercontent.com/chafey/dicomParser/master/dist/dicomParser.min.js)

**Dicomread:**

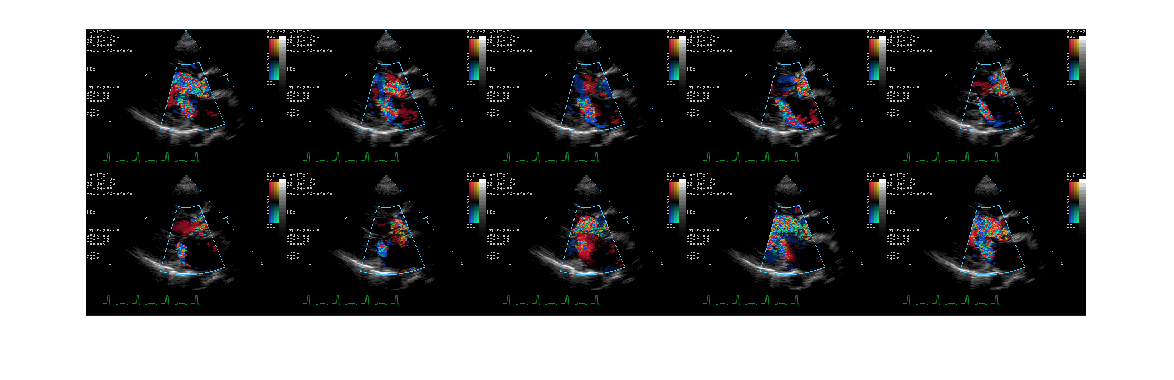
**Syntax:**

X = dicomread(filename)  
X = dicomread(info)  
[X,map] = dicomread(...)  
[X,map,alpha] = dicomread(...)  
[X,map,alpha,overlays] = dicomread(...)  
[...] = dicomread(filename,'frames',n)  
[...] = dicomread(**\_\_\_**,'UseVRHeuristic',TF)

Read indexed image from DICOM file and display it using montage.

[X, map] = dicomread('US-PAL-8-10x-echo.dcm');

montage(X, map, 'Size', [2 5]);



Read image from DICOM file and display it using imshow.

info = dicominfo('CT-MONO2-16-ankle.dcm');

Y = dicomread(info);

figure

imshow(Y,[]);



**Dicomwriter:**

**Syntax:**

dicomwrite(X, filename)  
dicomwrite(X, map, filename)  
dicomwrite(..., param1, value1, param2, value2, ...)  
dicomwrite(..., 'ObjectType', IOD,...)  
dicomwrite(..., 'SOPClassUID', UID,...)  
dicomwrite(..., meta\_struct,...)  
dicomwrite(..., info,...)  
status = dicomwrite(...)

Read a CT image from the sample DICOM file included with the toolbox.

X = dicomread('CT-MONO2-16-ankle.dcm');

Write the CT image to a file, creating a secondary capture image.

dicomwrite(X, 'sc\_file.dcm');

Write the CT image, X, to a DICOM file along with its metadata. Use the dicominfo function to retrieve metadata from a DICOM file.

metadata = dicominfo('CT-MONO2-16-ankle.dcm');

dicomwrite(X, 'ct\_file.dcm', metadata);

Copy all metadata from one file to another. When you set the 'CreateMode' parameter to 'copy', dicomwrite does not verify the metadata written to the file.

dicomwrite(X, 'ct\_copy.dcm', metadata, 'CreateMode', 'copy');