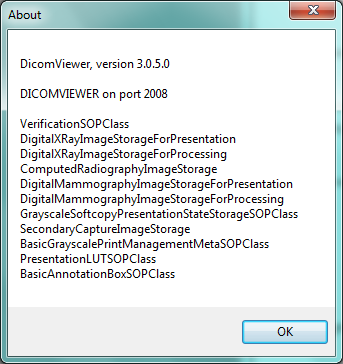
# EK.Capture.Dicom.DicomToolkit

The DicomToolkit is a DICOM library that supports nearly [[1]](#footnote-1)all the DICOM functionality that a modality like Carestream Health needs. The source code can be found in the IPA repository.

IPA will use the DicomToolkit for NextGen DICOM file reading and writing. The toolkit has support for DICOM service classes for workflow, storage, reporting and printing.

The DicomToolkit, along with a handy set of tools written using the library are included in the nightly build and are installed as part of the platform installation. This means that they are always available on any product machine that is based on the Capture Platform.

For those tools that can be used as a DICOM SCP, the Help... About menu item will show you the AE title and port that the server is listening on, along with the DICOM service classes that it supports.



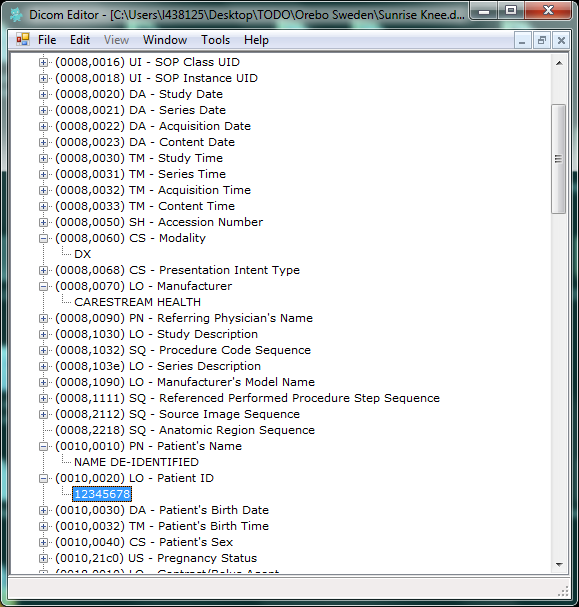
The dialog above shows that the DicomViewer listens on port 2008, and can serve as a printer or storage destination.

Some of the tools support automation via the command line to facilitate batch file handling.

The following is a brief introduction to some of the tools in the library.

## DicomEditor

DicomEditor can be used to create or edit DICOM part 10 files, DICOMDIRs and DICOM datasets.



DicomEditor can be used as a printer or storage simulator.

When in PACS mode, it can act as a storage destination that saves all images delivered to it in a DICOMDIR. It is capable of Storage Commitment and image archive retrieval. This means that it can be used to test Priors.

Some features of the DicomEditor can be automated from the command line. It can be used to create new DICOM files, or update individual tag values. This is used heavily by QA to generate and update DICOM files that are used in Modality Worklist testing.

A macro is a fragment of text that is surrounded by ‘?’. For example, ?NOW(-1d)?, would be replaced with today’s date minus one day.

|  |  |  |
| --- | --- | --- |
| Macro | Description | Notes |
| NOW(#) | Date, Time or DateTime. | Based on current time.  The tag’s VR determines the starting value: DA=Today's Date; TM=Current Time; DT=Current Date and Time.  The # argument modifies the DateTime by adding or subtracting based on a pattern +/-#y#m#d#h#n#s  Example: |
| UID | New UID. | Generates a new unique DICOM UID. |
| DATE | Random Date. | Any random Date |
| TIME | Random Time. |  |
| ALPHA(#) | Random Alpha string. |  |
| NUMERIC(#) | Random Numeric string. |  |
| ALPHANUMERIC(#) | Random Alphanumeric string. |  |
| ASCENDING(#) | An Ascending number. | Milliseconds since 2012 |
| NULL | Empty out a tag. |  |
| REMOVE | Remove a tag, if it exists. |  |
| Macros can be combined. | | |

## DicomViewer

The DicomViewer is used to view images. It supports DICOM files, raw files, and any .Net compatible file format. It uses the file extension in order to know what file format to use (dcm, raw, mask, bmp, jpg, tif, tiff, gif, png, emf, exif, and wmf).

You can open a file in one format and save it another.



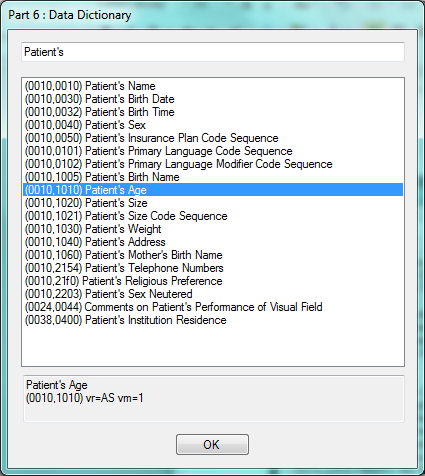
The DicomViewer can be used to put together the Eclipse image chain.

First open a raw processed file, then drag on

## DicomTags

DicomTags is a DICOM Data Dictionary lookup tool. You use it to look up DICOM tags.

The search box at the top uses an incremental string search to filter the DICOM Data dictionary for tags that match. You can filter on Descriptions or tag numbers.

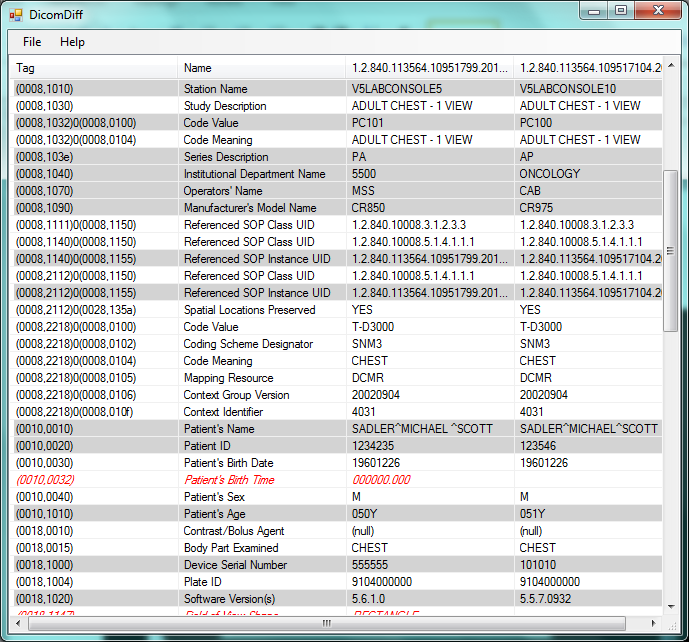


When you highlight an entry, you can see additional DICOM dictionary information at the bottom.

## DicomDiff

DicomDiff can be used to compare two DICOM part 10 files, or DICOM fragments.

All the tags from both files are displayed. If the tag values differ, the row is highlighted. If one file contains a DICOM tag that the other file does not, the line is shown in red.



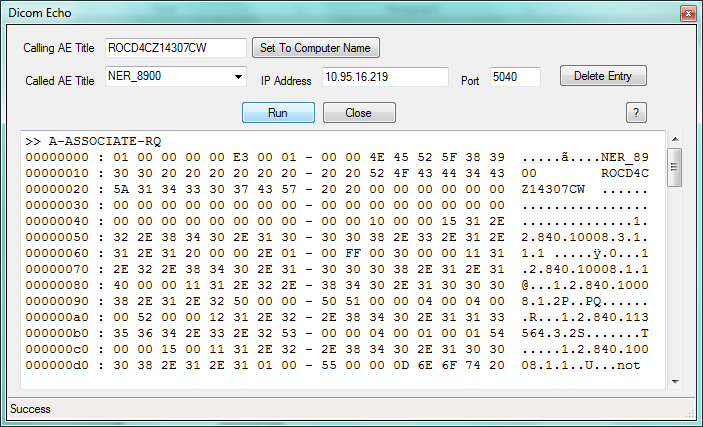
Binary data is also compared, however the data may not be shown depending on the amount of data in the tag.

A batch command line feature will be added so that DICOM file comparison can be automated.

## DicomEcho

DicomEcho is used to verify connectivity with a DICOM SCP. It can be used to debug a DICOM connection.

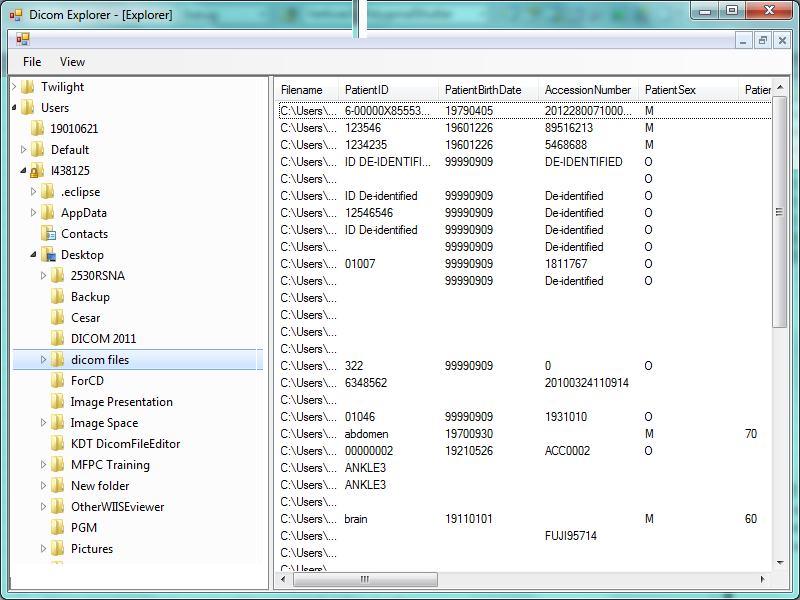
A binary dump of the conversation is written to the text box so that the DICOM can be examined. The status bar indicates status.



## DicomExplorer

DicomExplorer is used to browse the file system and show the selected contents of DICOM files.

This makes it easier to find the DICOM files that you are looking for without opening each file.



The tool has an export feature that can be used to export a list of what it finds. QA uses this for their whacky test plans.

## DicomPipe

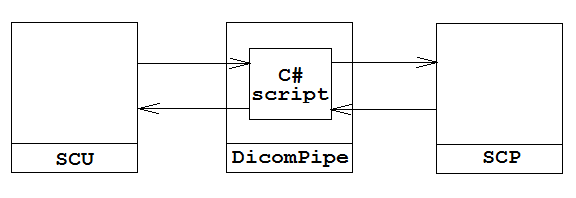
How do you get an otherwise well behaved PACS to violate the DICOM standard? How would you make a printer support a transfer syntax that it does not normally support?

You use DicomPipe.

First off, Why on Earth would you want to do this?

* QA often has a need to simulate a naughty PACS, or simulate something that is happening in the field that violates the DICOM standard.
* QA and Developers have a need to test software under error conditions.
* You may not have permission to install WireShark on a customer’s machine and you want to capture a DICOM session.

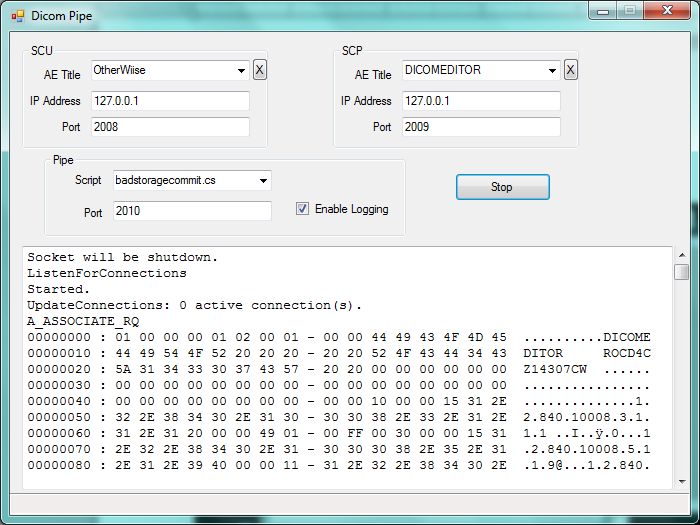
DicomPipe is an advanced tool that can be used to sit between an SCU and SCP and monitor or alter the DICOM that flows between the two end points via scripting.



DicomPipe sits in the middle. Beside configuring host information, the SCU and the SCP cannot tell the difference.

At a minimum it can be used to log the conversation between two DICOM devices. This can be useful if you cannot install WireShark on a production machine.

This tool needs some bug fixes.



Normally you only have to fill in the SCP information. In the cases where the SCU acts as an SCP, you will have to fill in the SCU information as well. Examples of this are Storage Commitment callbacks and responses to Image Archive C-MOVE commands.

### Scripting

Advanced users can write a script to alter the DICOM that is transferred back and forth. This could be used to do anything from simulating an SCP that violates the DICOM standard, to changing the transfer syntax passed between the two devices or altering any of the data before it is passed on to either side.

A specialized C# script can be written that is given access to each PDU before it is passed along. The script references the DicomToolkit, along with any other assemblies in the PATH.

The script needs to contain a public class named ‘Script’ that contains a public static method called ‘OnPdu’. OnPdu takes an EK.Capture.Dicom.DicomToolkit.ProtocolDataUnit as an argument. OnPdu returns a bool which should indicate whether or not the PDU was altered.

|  |
| --- |
| using System;  using EK.Capture.Dicom.DicomToolKit;  public class Script  {  public static bool OnPdu(EK.Capture.Dicom.DicomToolKit.ProtocolDataUnit pdu)  {  return false;  }  } |

Examples:

Delay: QA wanted to test the timeouts that were configurable for DICOM deliveries and how the software reacted to a noisy network.

They needed a very slow SCP that would take a long time to respond to DICOM commands. None of their normal DICOM destinations could be used to test this, and they could not create their own server for the testing.

The following script simply adds a half second delay to each step of the DICOM conversation. They were able to alter the script to simulate different delays.

|  |
| --- |
| using System;  using EK.Capture.Dicom.DicomToolKit;  public class Script  {  public static bool OnPdu(EK.Capture.Dicom.DicomToolKit.ProtocolDataUnit pdu)  {  // delay for half a second  System.Threading.Thread.Sleep(500);  return false;  }  } |

Error Condition: QA wanted to simulate a failed delivery status being sent by an SCP so that they could test how the software handled the failure.

The following script intercepts any response to a DICOM command and changes the status from success to error number 204.

|  |
| --- |
| using System;  using System.Windows.Forms;  using EK.Capture.Dicom.DicomToolKit;  public class Script  {  public static bool OnPdu(EK.Capture.Dicom.DicomToolKit.ProtocolDataUnit pdu)  {  switch (pdu.PduType)  {  case ProtocolDataUnit.Type.P\_DATA\_TF:  {  PresentationDataPdu message = pdu as PresentationDataPdu;  foreach (PresentationDataValue pdv in message)  {  if (pdv.Control == MessageType.LastDataSet)  {  DataSet command = pdv.Dicom;  if (command.Contains(t.Status))  {  ushort status = (ushort)command[t.Status].Value;  if (status == 0)  {  status = 204;  }  }  return true;  }  }  }  break;  }  // return true if you change the pdu, false if you did not  return false;  }  } |

# TraceLogViewer

# UnitTestDriver

1. The toolkit has limited support for Japanese and multi-frame images. [↑](#footnote-ref-1)