JDCM
Java DICOM toolkit

User's Guide

Overview

This user's guide is targeted toward the developer of medical imaging applications using the JDCM Tool Kit to supply DICOM network or media functionality. JDCM supplies you with a powerful and simplified interface to DICOM. It lets you focus on the important details of your application, rather than the often complex and confusing details of the DICOM Standard.

The DICOM standard

The DICOM *(Digital Imaging and Communications in Medicine)* Standard was originally developed by a joint committee of the American College of radiology *(ACR)* and the National Electrical Manufacturers Association *(NEMA)* to facilitate the open exchange of information between digital imaging computer systems in medical environments. DICOM Version 3.0 is composed of several hundreds of pages over sixteen separate parts. Each part of the standard focuses on a different aspect of the DICOM protocol.

```
The ACR/NEMA Standard Home Page
http://www.nema.org/

David Clunie's site (with links to the standard)
http://www.dclunie.com/
```

Developing DICOM applications

The JDCM Library is 100% pure Java TM. Provides classes that represent all of the major basic components of the DICOM.

DICOM Associations

- A_Abort
- A_Associate
- A_Release
- DCMServer

DICOM Messages and Message Elements

- DicomElement
- DicomSequence
- DicomGroup
- DicomSet
- DimseService

DICOM Network Service classes

- C_Echo
- C_Find
- C_Get
- C_Move
- C_Store
- N_Action
- N_Create
- N_Delete
- N_Event_Report
- N_Get
- N_Set

DICOM media services classes

- DicomFile

DICOM and Toolkit constants

- StaticProperties

Library import

In order to access the classes of the JDCM class library you should import the *jdcm.jar* package in your applications.

```
import jdcm.*;
```

Getting the Class Library Version

You can use the static `IMPLEMENTATION_VERSION` filed of the `StaticProperties` class to retrieve the string identifying the JDCM Library version. The library version string is of the form *JDCM-V-n.m.v* where *(n)* is major version number, *(m)* is minor version number and *(v)* is an interim release number.

Association Management

You will probably want to initiate an association if you are a client, or wait for an association if you are a server. Clients will use the Constructors of `A_Associate` class and servers will inherit the `DCMServer` class. Before you establish an association connection you must determine what DICOM services you are prepared to handle and perhaps create a list of proposed context to encapsulate the service information.

A proposed context object represents:

- One DICOM service (SOP class).
- A set of transfer syntaxes you can support for the service.
- Presentation Context ID assigned to context's service.

Making the connection

Make a TCP/IP connection with the peer DICOM entity making use of the standard *java.net* classes. In our example this is the server *localhost* at port number *104 (Default DICOM port)*.

```
Socket socket = new Socket("localhost",104);
```

- **Association sample 1**

In this example the application entity title of the DICOM application with which we want to connect is *ANY-SCP* and our own application entity title is *JDCM*. Propose a SOP Class with *Implicit Little Endian* syntax transfer.

```
A_Associate associate;
```

```
try {
    associate = new  A_Associate(
    socket,"JDCM",
    "ANY-SCP",
    "1.2.840.10008.5.1.4.1.1.6.1", //affected SOPClass
    "1.2.840.10008.1.2");          //transfert Syntax
.. .

} catch(IOException e) {...}
```

Message Exchange

DICOM messages are exchanged with other application entities over the network. The exchange of DICOM messages between applications only occurs over an open association. After the DICOM client *(SCU)* application opens an association with a DICOM server *(SCP)*, the client sends request messages to the server application. For each request message, the client receives back a corresponding response from the server. The server waits for a request message, performs the desired service, and sends back some form of status to the client in a response message.

Sending Network Messages

You must use a sub-class of the `DimseService` class: `C_Echo`, `C_Find`, `C_Get`, `C_Move`, `C_Store`, `N_Action`, `N_Create`, `N_Delete`, `N_Event_Report`, `N_Get`, `N_Set` to send messages and relate the instance with a specific association by passing an `A_Associate` reference as a parameter to the class constructor. `DimseServices` sub-classes require that you assign an Affected SOP Class UID to the composite message object being sent; in those cases, you must use the `setAffectedSOPClass()` method providing the UID.

```
C_Find find = new C_Find(associate);
find.setAffectedSOPClass(
StaticProperties.InformationModelFINDPatientRootQueryRetrieve);
```

To send request messages you use the writeRQ() method.

DimseService sub-classes require that you assign a reference to a DicomSet object that encapsulates the message to be sent.

```
//Dicom set encapsulate the message to be sent
DicomSet dicomSet = new DicomSet();
dicomSet.setElement(new DicomElement(0x0008,0x0052,"CS","PATIENT"));
dicomSet.setElement(new DicomElement(0x0010,0x0010,"PN","*"));
dicomSet.setElement(new DicomElement(0x0010,0x0020,"LO","*"));
dicomSet.setElement(new DicomElement(0x0010,0x0040,"CS","*"));
find.writeRQ(dicomSet);

do{
    dicomSet = find.readRPS();
    //retrieve the status of the response message
    status = find.getStatus();
} while(status != 0);
```

The getStatus() method retrieve the status of the requested operation and must be a valid response code for the service involved. Response codes for specific DICOM commands are described in Part 4 of the DICOM Standard.

Releasing or Aborting the Association

The DICOM standard requires the association requester to release the association when no further processing is required. This is done using the A_Release class.

```
A_Release release = new A_Release(associate);
release.writeRQ();
release.readRPS();
```

At any time either association partner may abort the association. This is used only in abnormal situations.

```
A_Abort abort = new A_Abort(associate);
abort.writeRQ();
```

After calling release or abort, close the association and socket

```
associate.close();
socket.close();
```

After no other methods should be called for associate object.

Starting an Association Acceptor

When an application acts as an association receiver, it waits for incoming *TCP/IP* connections with the accept() method from java.net.ServerSocket at a certain port number *(104)* and devotes a DCMServer thread to every incoming connection. You must provide a class that will process the new association. The class must extends the DCMServer class.

```
public class AppSCP extends DCMServer {

public AppSCP(Socket socket){
    super(socket);
    …
}

public static void main(String[] args){
    try {
        AppSCP        appSCP;
        Socket        socket;
        ServerSocket  serverSocket = new ServerSocket(104);

        while(true) {
            socket    = serverSocket.accept();
            appSCP  = new AppSCP(socket);
            appSCP.start();
        }
    } catch(Exception e){...}
}
}
```

If you need to know about the association your class requires override the DCMServer.A_Associate method that will be passed a newly created A_Associate object. When JDCM has successfully started an association, it will start a new thread and call your class's A_Associate() method. Everything you need to know about the association is provided in the A_Associate object.

```
public void A_Associate(A_Associate associate) throws IOException {
    associate.readRQ();
    callingEntity = associate.getCallingEntity();
    calledEntity  = associate.getCalledEntity();
    ...
}
```

Before DICOM messages can be exchanged across the association, the association acceptor must either accept or reject the association request from the association

requestor. If this application agrees with the acceptable services, it calls the `writeRPS()` method to establish an association between the two applications. If it disagrees, for some reason, it calls the `reject()` method.

```
public void A_Associate(A_Associate associate) throws IOException {
    associate.readRQ();
    callingEntity = associate.getCallingEntity();
    calledEntity  = associate.getCalledEntity();
    ...
    if (<the service is acceptable>)
       associate.writeRPS();
    else
        associate.reject(A_Associate.REJECTED_PERMANENT,1);
}
```

If you need process or know about a requested Dimse services: C-Echo, C-Store, C-Move, C-Find, C-Get, N-Action, N-Create, N-Delete, N-Event-Report, N-Get, N-Set or C-CancelRQ. Your class requires override the `DCMServer` methods.

```
public void C_Echo(DimseService dimseService)throws IOException
public void C_Find(DimseService dimseService)throws IOException
public void C_Get(DimseService dimseService)throws IOException
public void C_Move(DimseService dimseService)throws IOException
public void C_Store(DimseService dimseService)throws IOException
public void N_Action(DimseService dimseService)throws IOException
public void N_Create(DimseService dimseService)throws IOException
public void N_Delete(DimseService dimseService)throws IOException
public void N_Event_Report(DimseService dimseService)throws IOExce..
public void N_Get(DimseService dimseService)throws IOException
public void N_Set(DimseService dimseService)throws IOException
public void C_CancelRQ(DimseService dimseService)
```

The general pattern for a Dimse service method is to extract information from the request, process the information and then write the response. A request contains data passed between a client and the server. `DimseService` class defines methods for accessing the following information:

- Transfer Syntax Used
- Command Set properties: MessageID, Status, Prority
- References to message's command and message's data Set

StoreSCP acceptor sample:

```
public void C_Store(DimseService dimseService) throws IOException {

// Get data set received
DicomSet dicomSet = dimseService.readRQ();
try {

    // Write dicom file in hard disk
    ((C_Store) dimseService).writeFile("C:/receivedFile.dcm");
    // Set succesfully response
    dimseService.setStatus((short)0x0000);
} catch(IOException e) {
    //Set failed received stream response;
    dimseService.setStatus((short)0xA700);
}
// Write response
dimseService.writeRPS(null);
}
```

If you need to know about the release or abort request Your class requires override the DCMServer A_Release() method and the A_Abort() methods.

```
public void A_Release(A_Release release) throws IOException
public void A_Abort(A_Abort abort) throws IOException
```

Association Properties

The A_Associate class contains several methods that can be used to retrieve properties of the association.

- **Application Entity Title**

Each DICOM application is assigned an application entity ID, known also as the application title. The getCallingEntity() method retrieves the application title of the calling application and the getCalledEntity() method retrieves the application title of the called application.

- **Implementation Class UID and Implementation Version**

The identification of an implementation of the DICOM standard relies on two pieces of information: the Implementation Class UID (required) and the Implementation Version Name (optional). The DICOM standard requires that association requestors and acceptors notify each other of their respective

Implementation ClassUID. The `getCalledImplementationClassUID()` method returns the Implementation Class UID of the local application and the `getCalledImplementationVersion()` method returns the Implementation Version of the called application. The `getCallingImplementationClassUID()` method returns the Implementation Class UID of the calling application and the `getCalledImplementationVersion()` method returns the Implementation Version of the called application.

- **Accepted presentation context**

During association negotiation the client and the remote DICOM system exchanged the list of presentation supported and accepted. `GetAcceptedPresContext()` retrieve the list of presentation context accepted

- **Maximum PDU Sizes**

During association negotiation the client and the remote DICOM system exchanged the maximum size of Protocol Data Units that each is willing to receive. Each system commits to send package data no larger than that negotiated for the receiver. The `getCalledMaximunLength()` returns the size of the largest PDU that the called system is willing to receive. The `getCallingMaximunLength()` method returns the size of the largest PDU that the calling system is willing to receive. To set maximun PDU Size in association negotiation use `StaticProperties.MAXIMUN_LENGTH` default 16384

DimseMessage Properties

The `DimseService` class contains several methods that can be used to retrieve or set properties of the DIMSE message: C-FIND, C-MOVE, C-STORE, C-ECHO, N-ACTION, N-CREATE, N-DELETE, N-EVENT-REPORT, N-SET, N-GET

- **Transfer Syntax Used**

The `getSyntaxTransfer()` method returns an String object that identifies the transfer syntax used to encode this message.

- **Contained commands and message's dataSets**

`DimseService` class contains references to message's command and message's dataSet. The `getCommandRPS()` and `getCommandRQ()` method returns the commands requests and responses references and the `getDicomSet()` method returns the data Set reference.

- **Command Set Properties**

The `getAffectedSOPClass()` method returns the DICOM Affected SOP Class UID associated with the DIMSE message. It is retrieved from attribute `(0000,0002)` in the message's command set. The `setAffectedSOPClass()` method sets the value of the DICOM Affected SOP Class UID associated with the DIMSE message.

- **MessageID**

The DIMSE service provider *(ex JDCM)* assigns a number to each DIMSE request message. The `getMessageID()` method retrieves that identifying number from the command set attribute `(0000,0110)`.

- **Status**

The `setStatus()` and `getStatus()` methods set and retrieves the response status code `(0000,0900)` from the message command response set.

- **Priority**

The `getPriority()` method retrieves the priority number from the command request set attribute `(0000,0700)`.

- **Presentation context ID**

The `getPresentationContextID()` method retrieves the DICOM Presentation Context ID assigned to this context's service for the current association.

Manipulation of attributes

The different attributes of DICOM data elements are internally stored in the `DicomElement` class. The `DicomElement` class contains a *(group, element)* pair and

the values of the attribute stored. The Java type that represents the values of an attribute depends on the DICOM type *(Value Representation)* defined for that specific attribute. The following table shows the mapping between a DICOM type and its corresponding Java type.

DataType of value	May be used to set attributes with these VRs
int	SL, UL, US, SS, IS
double	FL, FD, OF
String	AE, AS, AT, CS, DS, LO, LT, PN, SH, SS, ST, UI, UN, UT, DT, DA, TM, AT
byte[]	OB, OW
Sequence	SQ, OB, OW

int

```
DicomElement(int group, int element,String vr, int value)
DicomElement(int group, int element,String vr, int[] value)
getValueAsInt(int index)
getValueAsInt()
```

double

```
DicomElement(int group, int element,String vr, int double)
DicomElement(int group, int element,String vr, int[] double)
getValueAsDouble(int index)
getValueAsDouble()
```

String

```
DicomElement(int group, int element,String vr,String value)
DicomElement(int group, int element,String vr,String[] value)
getValueAsString(int index)
getValueAsString()
```

byte[]

```
DicomElement(int group, int element, String vr, byte[] value)
getValue()
```

Sequence

```
DicomElement(int group, int element,String vr, DicomSequence sequence)
getSequence()
```

Data Sets and Data groups

In JDCM all DICOM data sets are represented by DicomSet class. This class is a subclass of java.util.Vector. Basically this is a container of DicomGroup class. DicomGroup s may be retrieved or added to an data Set using the getGroup() or setGroup() methods of DicomSet class. Data Elements may be retrieved or added to an data set using the getElement() or setElement() methods of DicomSet class *(see API)*.

```
// Constructing Data Sets
DicomSet dicomSet = new DicomSet();
dicomSet.setElement(new DicomElement(0x0008,0x0052,"CS","PATIENT"));
dicomSet.setElement(new DicomElement(0x0010,0x0010,"PN","*"));
dicomSet.setElement(new DicomElement(0x0010,0x0020,"LO","*"));
dicomSet.setElement(new DicomElement(0x0010,0x0040,"CS","*"));
```

For users who want to manipulate groups there are a DicomGroup class. A group is a sub-class of java.util.Vector. It is a set of DicomElement which have the same group number in their *(group, element)* pair. Data Elements may be retrieved, added or removed from group using the getElement(), setElement() removeElement() methods *(see API)*.

Sequences of Items

The DICOM Value Representation SQ is used to indicate a DICOM attribute that contains a value that is a sequence of items. Each item in the sequence is a data set. Each of the data sets can also contain attributes that have a VR of SQ.

```
DicomSet dicomSet1 = ...;
DicomSet dicomSet2 = ...;
DicomSet dicomSet3 = ...;

ArrayList aList= new ArrayList();
aList.add(dicomSet1);
aList.add(dicomSet2);
aList.add(dicomSet3);

//if undefined is true length of sequence is undefined and
//Sequence Delimitation Item is added.
boolean undefined = true;
DicomSequence dicomSequence = new DicomSequence(aList,undefined);
```

```
DicomElement dicomElement = new
DicomElement(0x0040,0xa043,"SQ",dicomSequence);
```

Encapsulated Pixel Data

The DicomSequence class handle the sequence of pixel Data Element (0x7FE0,0x0010) in encapsulated format.

```
byte[] array1 = ...;
byte[] array2 = ...;
byte[] array3 = ...;

ArrayList aList= new Vector();
aList.add(array1);
aList.add(array2);
aList.add(array3);
DicomSequence dicomSequence = new DicomSequence(aList);
DicomElement dicomElement =
new DicomElement(0x7fe0,0x0010,"OB",dicomSequence);
```

DICOM Files objects

DICOM media files are encapsulated in the DicomFile class. The DicomFile class contains a special file meta information getMetaFile() and a data set getDicomSet().

Reading DICOM Files

There are two options available to construct new instances of the DicomFile class associated with a specified file system fileName.

```
try {
    DicomFile dicomFile = new DicomFile("C:/dcmFile.dcm");
    ...
} catch(IOException e){...}

try{
    File file = …
    DicomFile dicomFile = new DicomFile(file)
} catch(IOException e){   }
```

Working with the contained file meta information

The File Meta Information encapsulated in the DicomFile class contains identifying information about the data set also encapsulated in a DICOM file. The meta information consists of a fixed-length 128-byte file Preamble, a DICOM Prefix *(DICM)*, followed by several DICOM attributes providing the properties of the encapsulated data set. The contents of this object are maintained automatically by JDCM, although the getMetaFile() method of the DicomFile class returns a reference to its contained DicomGroup object.

```
try{
    //Constructing a DicomFile Instance
    DicomFile dicomFile = new DicomFile("C:/dicomFile.dcm");
    //Get the file meta info attribute set
    DicomGroup dicomGroup = dicomFile.getMetaFile();

    DicomElement dicomElement = null;
    for(Enumeration e = dicomGroup.elements();e.hasMoreElements() ;) {
        dicomElement = (DicomElement)e.nextElement();
        System.out.println(dicomElement);
    }
} catch(){...}
```

The data set DicomSet object encapsulated in the DicomFile class contains the DICOM information object associated with the file. You can retrieve a reference to the contained DicomSet object with the getDicomSet() method. With the getDicomElements() method of DicomSet class it is possible to get an iterator of all attributes contained in a DicomSet object.

```
//Get the data set encapsulated in the DicomFile
DicomSet dicomSet = dicomFile.getDicomSet();

DicomElement dicomElement;
for (Iterator i = dicomSet.getDicomElements(); e.hasNext ();) {
    dicomElement = (DicomElement)e.next ();
    System.out.println(dicomElement);
}
```

Creating and Writing DICOM Files

To construct a DicomFile object from a DicomSet and writes it using specific transfer syntax.

```
try {
    DicomSet ds = new DicomSet();
    ds.setElement(new DicomElement(0x0008,0x0008,"UI",
                  new String[] {"DERIVED","SECONDARY"}));
    ds.setElement(new DicomElement(0x0008,0x0016,"UI",
                  "1.2.840.10008.5.1.4.1.1.7"));

    //prepare pixel data
    byte[] raw = new byte[640*480];
    for (int i = 0; i < 640;i++)
        for (int j = 0; j< 480;j++)
            raw[j*640+i] = (byte)(i + j);

    ds.setElement(new DicomElement(0x7fe0,0x0010,"OB",raw));

    DicomFile dicomFile = new DicomFile(ds);
    dicomFile.write("D:/Anoymous.dcm",
                    StaticProperties.TS_LITTLE_ENDIAN_EXPLICIT);

} catch(IOException e){}
```

Dictionary

JDCM package load DICOM data dictionary from *jdcm/dictionary.properties* in *jdcm.jar* file. You can extend the default dictionary with your private dictionary using.

```
StaticProperties.setPrivateDictionary("private.dic");
```

Private dictionary sample:

```
0019,0013=CRImageIPParamsRight,ST,1
0029,0050=SceneText,LT,1
00E1,0040=OffsetFromCTMRImages,SH,1
0019,002f=TriggerFrequency,DS,1
```

Accepted presentation context

JDCM package load accepted presentation context and transfer syntax from jdcm/presContext.properties in jdcm.jar file. You can extend the default presentation context with your preferred configuration using.

```
StaticProperties.setPresContext("presContext.txt")
```

Library Constants

Setting/getting properties in `StaticProperties` class.

AETitle (default: JDCM)	AET
Implementation Class	UID IMPLEMENTATION_CLASS
Implementation version	IMPLEMENTATION_VERSION
Verbose (default false)	VERBOSE
Max PDU Size (default 16384)	MAXIMUN_LENGTH
Organization root UID	ORG_ROOT

Conformance statement

Since JDCM class library is not an application, this conformance statement only gives an outline of the of the DICOM services it supports:

▸ Application data flow diagram

JDCM provides the core functionality required to facilitate data flow between SCUs and SCPs.

▸ AE specifications

JDCM provides support for applications, which deal with the following Service Object Pairs (SOPs):

1.2.840.10008.1.1	Verification SOP Class

JDCM default configuration, supports the following Storage SOP Classes as an SCU/SCP:

1.2.840.10008.5.1.1.27	StoredPrintStorage
1.2.840.10008.5.1.1.29	HardcopyGrayscaleImageStorage
1.2.840.10008.5.1.1.30	HardcopyColorImageStorage
1.2.840.10008.5.1.4.1.1.1	ComputedRadiographyImageStorage
1.2.840.10008.5.1.4.1.1.1.1	DigitalXRayImageStorageForPresentation
1.2.840.10008.5.1.4.1.1.1.1.1	DigitalXRayImageStorageForProcessing
1.2.840.10008.5.1.4.1.1.1.2	DigitalMammographyXRayImageStorage ForPresentation
1.2.840.10008.5.1.4.1.1.1.2.1	DigitalMammographyXRayImageStorage ForProcessing
1.2.840.10008.5.1.4.1.1.1.3	DigitalIntraOralXRayImageStorage ForPresentation
1.2.840.10008.5.1.4.1.1.1.3.1	DigitalIntraOralXRayImageStorage ForProcessing

1.2.840.10008.5.1.4.1.1.2	CTImageStorage
1.2.840.10008.5.1.4.1.1.3.1	UltrasoundMultiframeImageStorage
1.2.840.10008.5.1.4.1.1.4	MRImageStorage
1.2.840.10008.5.1.4.1.1.4.1	EnhancedMRImageStorage
1.2.840.10008.5.1.4.1.1.4.2	MRSpectroscopyStorage
1.2.840.10008.5.1.4.1.1.6.1	UltrasoundImageStorage
1.2.840.10008.5.1.4.1.1.7	SecondaryCaptureImageStorage
1.2.840.10008.5.1.4.1.1.7.1	MultiframeSingleBitSecondaryCapture ImageStorage
1.2.840.10008.5.1.4.1.1.7.2	MultiframeGrayscaleByteSecondaryCapture ImageStorage
1.2.840.10008.5.1.4.1.1.7.3	MultiframeGrayscaleWordSecondaryCapture ImageStorage
1.2.840.10008.5.1.4.1.1.7.4	MultiframeTrueColorSecondaryCapture ImageStorage
1.2.840.10008.5.1.4.1.1.8	StandaloneOverlayStorage
1.2.840.10008.5.1.4.1.1.9	StandaloneCurveStorage
1.2.840.10008.5.1.4.1.1.9.1.1	TwelveLeadECGWaveformStorage
1.2.840.10008.5.1.4.1.1.9.1.2	GeneralECGWaveformStorage
1.2.840.10008.5.1.4.1.1.9.1.3	AmbulatoryECGWaveformStorage
1.2.840.10008.5.1.4.1.1.9.2.1	HemodynamicWaveformStorage
1.2.840.10008.5.1.4.1.1.9.3.1	CardiacElectrophysiologyWaveformStorage
1.2.840.10008.5.1.4.1.1.9.4.1	BasicVoiceAudioWaveformStorage
1.2.840.10008.5.1.4.1.1.10	StandaloneModalityLUTStorage
1.2.840.10008.5.1.4.1.1.11	StandaloneVOILUTStorage
1.2.840.10008.5.1.4.1.1.11.1	GrayscaleSoftcopyPresentation StateStorage
1.2.840.10008.5.1.4.1.1.12.1	XRayAngiographicImageStorage
1.2.840.10008.5.1.4.1.1.12.2	XRayFluoroscopyImageStorage
1.2.840.10008.5.1.4.1.1.20	NuclearMedicineImageStorage
1.2.840.10008.5.1.4.1.1.66	RawDataStorage
1.2.840.10008.5.1.4.1.1.77.1.1	VLEndoscopicImageStorage
1.2.840.10008.5.1.4.1.1.77.1.2	VLMicroscopicImageStorage
1.2.840.10008.5.1.4.1.1.77.1.3	VLSlideCoordinatesMicroscopic ImageStorage
1.2.840.10008.5.1.4.1.1.77.1.4	VLPhotographicImageStorage
1.2.840.10008.5.1.4.1.1.88.11	BasicTextSR
1.2.840.10008.5.1.4.1.1.88.22	EnhancedSR
1.2.840.10008.5.1.4.1.1.88.33	ComprehensiveSR
1.2.840.10008.5.1.4.1.1.88.50	MammographyCADSR
1.2.840.10008.5.1.4.1.1.88.59	KeyObjectSelectionDocument
1.2.840.10008.5.1.4.1.1.128	PETImageStorage
1.2.840.10008.5.1.4.1.1.129	PETCurveStorage
1.2.840.10008.5.1.4.1.1.481.1	RTImageStorage
1.2.840.10008.5.1.4.1.1.481.2	RTDoseStorage
1.2.840.10008.5.1.4.1.1.481.3	RTStructureSetStorage
1.2.840.10008.5.1.4.1.1.481.4	RTBeamsTreatmentRecordStorage
1.2.840.10008.5.1.4.1.1.481.5	RTPlanStorage
1.2.840.10008.5.1.4.1.1.481.6	RTBrachyTreatmentRecordStorage
1.2.840.10008.5.1.4.1.1.481.7	RTTreatmentSummaryRecordStorage

Accepted transfer syntaxes for sending or receiving storage objects:

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
1.2.840.10008.1.2.2	Explicit VR Big Endian
1.2.840.10008.1.2.1.99	Deflated Explicit VR Little Endian
1.2.840.10008.1.2.4.50	JPEG Baseline (Process 1): Default Transfer Syntax for Lossy
1.2.840.10008.1.2.4.51	JPEG Extended (Process 2 & 4): Default Transfer Syntax for Lossy JPEG 12
1.2.840.10008.1.2.4.57	JPEG Lossless, Non-Hierarchical (Process 14)
1.2.840.10008.1.2.4.70	JPEG Lossless, Non-Hierarchical,
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
1.2.840.10008.1.2.5	RLE Lossless
1.2.840.10008.1.2.4.100	MPEG2 Main Profile @ Main Level

JDCM supports the compression and decompression of:

Deflated Explicit VR Little Endian 1.2.840.10008.1.2.5

As a *Query/Retrieve SCU/SCP*, JDCM default configuration supports the Find, Get and Move SOP classes for all standard information models:

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
1.2.840.10008.5.1.4.1.2.3.2	Patient/Study Only Query/Retrieve Information Model - MOVE
1.2.840.10008.5.1.4.1.2.3.3	Patient/Study Only Query/Retrieve Information Model - GET
1.2.840.10008.5.1.4.31	Modality Worklist Information Model – FIND
1.2.840.10008.5.1.4.32.1	General Purpose Worklist Information Model – FIND

Accepted transfer syntaxes for sending or receiving query/retrieve objects:

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
1.2.840.10008.1.2.2	Explicit VR Big Endian

Support for DIMSE-C services: C_Store, C_Get, C_Move, C_Find, C_Echo.

Support for DIMSE-N services: N_Action, N_Create, N_Delete, N_Event_Report, N_Get, N_Set.

All VR's are supported.

ACSE User Information

Maximum Length application PDU notification is supported
Implementation identification notification is supported
Asynchronous Operations Window is not supported
SCP/SCU role selection negotiation is supported
SOP Class Extended Negotiation is not supported
SOP Class Common Extended Negotiation is not supported
User Identity Negotiation is not supported

Release 1.6 Jun 2006
Release 1.5 April 2005
Release 1.4 Sep 2004
Release 1.3 July 2004

If you have suggestion, need technical information or find bugs with this software we would appreciate hearing about them. Please send electronic mail to: gigiobb@yahoo.com. Please try to describe the problem in detail.