|  |
| --- |
| Syngo.share/Biobank |
| Integration Documentation |

Contents

[Introduction 3](#_Toc384298336)

[Overview 3](#_Toc384298337)

[Detailed System Process 4](#_Toc384298338)

[Databases 6](#_Toc384298339)

[THE JAVA CRONS AND THEIR PROCESS DESCRIPTIONS 9](#_Toc384298340)

[CRON runtimes 9](#_Toc384298341)

[CRON config 9](#_Toc384298342)

[The CRON bash scripts 9](#_Toc384298343)

[Content of the bash script CRON files 9](#_Toc384298344)

[QUERY DICOM 11](#_Toc384298345)

[DECRYPT DATABASE 12](#_Toc384298346)

[UPLOAD CLINICAL 13](#_Toc384298347)

[QUERY ICE QUESTIONNAIRES 15](#_Toc384298348)

[RETRIEVE DICOM 16](#_Toc384298349)

[WEB SITE 17](#_Toc384298350)

[APPENDIX A 18](#_Toc384298351)

[Definitions 18](#_Toc384298352)

[APPENDIX B 19](#_Toc384298353)

[The clinical information from ICE\_LIVE 19](#_Toc384298354)

[APPENDIX C 20](#_Toc384298355)

[Java notes 20](#_Toc384298356)

[APPENDIX D 21](#_Toc384298357)

[Syngo.Share PACS 21](#_Toc384298358)

[APPENDIX E 22](#_Toc384298359)

[Upgrading the syngo.share library 22](#_Toc384298360)

[APPENDIX F 23](#_Toc384298361)

[Modality abbreviations 23](#_Toc384298362)

[APPENDIX G 24](#_Toc384298363)

[Functional Testing 24](#_Toc384298364)

[General functionally shared between all java processes 24](#_Toc384298365)

[queryDicom functional test table 24](#_Toc384298366)

[queryICEQuestionnaires functional test table 24](#_Toc384298367)

[retrieveDicom functional test table 25](#_Toc384298368)

[uploadClinical functional test table 25](#_Toc384298369)

[web site functional test table 25](#_Toc384298370)

[TEST MATRIX 26](#_Toc384298371)

[The tests 27](#_Toc384298372)

[Initial state before each test 27](#_Toc384298373)

[Test 1 - Initial Error testing 27](#_Toc384298374)

[Expected Results 27](#_Toc384298375)

[Test 2 - Initial state testing 27](#_Toc384298376)

[Expected Results 27](#_Toc384298377)

[Test 3 – No Match testing 27](#_Toc384298378)

[Expected results: 28](#_Toc384298379)

[Test 4 – queryDicom and queryICEQuestionaires testing 28](#_Toc384298380)

[Expected results: 28](#_Toc384298381)

[Test 5 – retrieveDicom testing 28](#_Toc384298382)

[Expected results: 29](#_Toc384298383)

[Test 6 – uploadClinical testing 29](#_Toc384298384)

[Expected results: 29](#_Toc384298385)

[Test 7 – web site testing 30](#_Toc384298386)

[Expected results: 30](#_Toc384298387)

[APPENDIX H 31](#_Toc384298388)

[A few possible problems and their solutions 31](#_Toc384298389)

[Unable to match a study to its BioBank ID, someone typed it in wrong, the web site is broken so we can’t manually match the study. 31](#_Toc384298390)

[There is a study which is generating problems, how do I tell the system to ignore it. This may happen if say the DICOM is deleted off the PACS before it’s processed (This will never happen in real life, of course). 31](#_Toc384298391)

[They have a PSI and we have not uploaded the clinical info, they want it uploaded NOW! How do I do it manually? 31](#_Toc384298392)

[Stuff isn’t working. What should I do? Where should I start looking? 32](#_Toc384298393)

## SYSTEM STATUS NOTES AS ON 04/04/2014

The process which downloads DICOM data from the PACS Cron job will be left disabled, as we aren’t archiving the data. Archiving into biobank is yet to be written.

Last name is now being placed in gtyp 61 so we can pick it up. Alan’s code is only in cvs, it hasn't been released to the testers let alone sent to the clinic.

## Introduction

The system and processes in this document covers the CTSU part of the management of DICOM data generated from the greater BioBank study, how this data is processed and stored in the BioBank archive.

## Overview

The first task is to collect all this generated data at one location. Luckily the devices used in the BioBank study which generate DICOM files already have the functionality to push that data to a central server running a PACS system. Once in this PACS system, the DICOM is checked over by a member of staff where they add keywords to help future research and categorisation.

After which point our processes come into play, we have automated the monitoring of this central system, downloading both the attached keywords and DICOM files. We are able to match up this medical data to the real BioBank participants during processing and storage of these files into the BioBank archive. While we are doing this we also detect scans which have been identified as a possible serious incidental finding, these require further assessment by an expert. In these cases we extract anonymised clinical information for that participant and upload it to the study in question in the PACs, the study can then be sent for further examination.

See **Appendix A** for definitions

## Detailed System Process

BioBank devices that generate DICOM will export it from the clinic to the central BIOBANK PACS running sygo.share. They will perform the export using the DICOM standard SCP/SCU transfer mechanism, which both the devices and PACS support. This part of the process will require no input from CTSU.

All DICOM modalities entering the PACS get stored in AU “BIOBANK\_QA”. For the first 1000 scans, a copy will also be automatically placed in AU “IF\_FIRST1K”; the purpose of this copy is so they can have a second opinion when checking for PSIs.

While still on the MRI scanner, the scans will be checked over; at this point a possible PSI could be raised. The MRI radiographer will raise this issue and get it reported on by the Lead Radiographer. If unable they will copy it straight into the AU “BIOBANK\_PSI” and bring it to the attention of the Incidental Findings Administrator. If confirmed as a PSI by the lead radiographer uploads a description of the findings to the problem scan.

In any case, PSI or not, the scan is automatically uploaded to AU “BIOBANK\_QA”.

All modalities in AU “BIOBANK\_QA” will be quality checked by a member of staff, tagging each series within a study with the appropriate keyword(s). Once performed the study is moved to AU “BIOBANK”. These QA keywords will be used to by future researchers to define their required image quality when selecting data for research. Only scans which are not at all fit for use will be removed off the system at this point. A further check is made that there is the appropriate DICOM image count for that type of series, if there is not then this is rectified by pushing the missing data from the scanner or it is flagged as incomplete with a keyword. As on the MRI scanner, if a PSI is noticed, a copy of this data will be sent to AU “BIOBANK\_PSI” for further investigation*.*

An automatic process will pick up on all new scans moved to AU “BIOBANK” and automatically download the data and their attached keywords. Once the data is situated on the local server the DICOM is grouped by study and sub grouped by series. The files are scanned and a manifest file is generated, the manifest and its corresponding series are compressed into a single file in preparation for archiving within the BioBank storage system.

Our monitoring process will also pick up on any scans copied into “BIOBANK\_PSI”, once found another process uploads that participant’s clinical information to the case. This information is taken from their BioBank touchscreen questionnaire. No participant identifiable details are uploaded.

The Incidental Findings Administrator monitors AU “BIOBANK\_PSI”, once the participant clinical information details are in place, it is moved to an appropriate Review Radiologist’s AU to be reported on. The Incidental Findings Administrator will alert the Review Radiologist to new cases as required.

The Review Radiologist will tag and upload any findings to a report and move the lot to AU “REPORTED”, where the appropriate action is taken by the Incidental Findings Administrator. If no serious medical incident is present, it will be moved to AU “COMPLETED”. If a serious medical incident is confirmed, then the case is escalated. Two letters are posted, one to the participant, asking him or her to make an appointment with their GP and a further one to alert their GP. Both will contain information on how to request CD’s of the relevant images.

Once letters are posted, the Incidental Findings Administrator moves the DICOM scan to AU “COMPLETED”.

There are various SOPs which cover this in more detail. The ones I know about are:

CLI82 - Imaging Incidental Findings SOP

CLI83 - Reporting Incidental Findings for UK Biobank SOP

# Databases

The system uses four databases, all which live on BB5:

* ace\_ice\_live
* ace\_ice\_trove\_live
* extract\_pacs\_live
* extract\_pacs\_ddb

The databases ace\_ice\_live and ace\_ice\_trove\_live are used to match up the random strings used as identifiers in the DICOM to the anonymous participant identifier used in the greater BioBank system. This part of the process uses extract\_pacs\_ddb, a distributed database for locating unprocessed entries. See Appendix B for more information on the ICE database. Once we have the BioBank identifier we can extract the BioBank questionnaire answers/questions as required to generate the clinical information.

The extract\_pacs\_live database is where all the data is stored for the running of this system. It was created and is accessed by the user account biocore.

The extract\_pacs\_live database contains the following tables:

|  |  |
| --- | --- |
| DICOMBIOBANK  When a series is downloaded from the syngo.share, the details are stored in this table. The information stored in this will also be used by the Biobank | |
| patientid | The DICOM participant id |
| studyid | DICOM study id. Retrieved from the DICOM file(s) |
| studyid\_enc | Encrypted dicombiobank.studyid |
| seriesid | DICOM series id. Retrieved from the DICOM file(s) |
| seriesid\_enc | Encrypted dicombiobank.seriesid |
| studydate | The date the study was inserted into the PACS |
| statusenc | 1 if we have not decrypted the encrypted fields, 0 if decrypted. Used by the decryption CRON job |
| zipmd5 | MD5 of the zipped DICOM study |
| studypk | The syngo.share primary key of the study |
| seriespk | The syngo.share primary key of the series |
| orginalsize | The pre-zip size of the DICOM series (bytes) |
| compressedsize | The size of the zip (bytes) |
| importstatus | Set to 1 once the BioBank import table has been filled |
| status | 1 if OK, anything else invalidates this row |
| numImages | The number of images in this Series |
| modality | The modality(s) in this series, see the DICOM standard for meaning |
| aet | The name of the device which generated this series |
| aesvi | The VI of the AES for this row |
| filename | Series Zip file name. Format: Modality\_trid\_cid\_SeriesPK\_StudyPK.zip |
| seriesdescription | The description of the series, taken from the DICOM |

|  |  |
| --- | --- |
| LOGS  Java errors and general information is stored in this table, which can be viewed using the website | |
| id | Primary Key |
| inserttimestamp | Timestamp |
| loglevel | 0=DEBUG,1=INFO,2=ERROR,3=CRITIAL,4=ACTIONREQUIRED |
| logtype | 0=GENERAL,1=DATABASE,2=FILESYSTEM,3=PACS |
| patientid | The DICOM patient id, if applicable |
| studydate | The Study date, if applicable |
| status | 1 if OK, anything else invalidates this row |
| message | The log message |

|  |  |
| --- | --- |
| DICOMDOWNLOADER  Keeps track of the last time the PACS was scanned for new DICOM entries | |
| lastdownload | Timestamp |

|  |  |
| --- | --- |
| BIOBANKPARTICIPANT  As BioBank questions are submitted into ICE, a CRON job extracts some of the data and places in this table | |
| pid | The pid of the Patient |
| dicomparticipantid | DICOM Participant id for matching |
| firstnameid | DICOM FirstName id for matching |
| cid | From the ICE row |
| trid | From the ICE row |
| scandate | The date it was entered into ICE |

|  |  |  |  |
| --- | --- | --- | --- |
| DICOMBIOBANK\_KEYWORD  A table to link together which keywords belong to which series | | |  |
| seriespk | FK into table DICOMBIOBANK |  | |
| keywordpk | FK into table KEYWORD |  | |

|  |  |
| --- | --- |
| CRONSTATUS  A table to keep track of when each CRON job starts and completes. It is displayed in the web site overview | |
| name | Name of the CRON |
| started | Timestamp of last time it started |
| finished | Timestamp of the last time it completed |

|  |  |
| --- | --- |
| STAFF  Holds the user’s sign on usernames who can access the web site | |
| name | The sign in account name allowed to use the web site |
| status | 1 if OK, anything else, ignores this user |

|  |  |
| --- | --- |
| PACSSTUDY  Every new study found on the PACS in the predefined AUs, get stored in this table | |
| patientid | The patientid found in the DICOM file |
| firstname | The firstname found in the DICOM file |
| confirmedpatientid | An entry here means we have an automatic match on either patientid or firstname into the ICE system or a manually one was entered by hand |
| studyuid | The DICOM studyid |
| studyuid\_enc | Encrypted version of studyid |
| studydate | The date the study was inserted into the PACS |
| aet | The name of the device which send study to the PACS |
| studypk | The syngo.share study primary key |
| modalities | The modalities syngo.share reports which are in this study |
| au | The AU this study is situated |
| insert | Timestamp this was inserted |
| update | Timestamp this was last updated |
| status | Invalid =0, OK=1, Complete=2, Ignore = 3 |
| patientpk | The syngo.share patient primary key |
| statusenc | 1 if we have not decrypted the encrypted fields, 0 if decrypted. Used by the decryption cron job |
| aesvi | The VI of the AES for this row |

|  |  |
| --- | --- |
| KEYWORD  Where details of the series level keywords attached to DICOM is stored | |
| keywordpk | Syngo.share keyword’s Primary Key |
| keyword | The text of the keyword |
| type | Type of this keyword, taken from syngo.share |
| status | 1 if OK, anything else invalidates this row |

# THE JAVA CRONS AND THEIR PROCESS DESCRIPTIONS

The task of keeping track of what is going on in the syngo.share system and performing the required actions is split up into 5 processes:

1. queryICEQuestionnaires.jar
2. queryDicom.jar
3. retrieveDicom.jar (turned off for now)
4. uploadClinical.jar
5. decryptdb.jar (turned off for now)

Each process is Java command line programs which are independently run as a CRON job. Each CRON job will be run periodically. On start-up each process has a mechanism to detect if it has another copy still running, in which case this is detected and it safely shuts down before it starts working.

Compiling note: At the top of database.java is a Boolean called LINUX. Check it is false when running on your local machine and true for bb5. You will also need to have your vnodes the same or change them in the code to reflect your local vnode setup.

There is also a web site for monitoring the system and can help a user do some manual interventions. This is a CGI based site, written in C++.

### CRON runtimes

1. queryICEQuestionnaires.jar – 5:00am, 1:00pm, 4:00pm,8:00pm: Monday - Saturday
2. queryDicom.jar – 5:15am,1:15pm,4:15pm, 8:15pm: Monday - Saturday
3. retrieveDicom.jar – (was 5:30am,1:30pm,4:30pm, 8:30pm: Monday – Saturday)
4. uploadClinical.jar – 1:45pm, 9:45pm: Monday - Saturday
5. decryptdb.jar - TBC

### CRON config

To get a copy of the current CRON config file, sign on to bb5 with biocore.

* “crontab –l > filename”

To push it back:

* “crontab filename”

### The CRON bash scripts

The cron jobs call a bash script with the same name as the jar.

### Content of the bash script CRON files

#! /bin/bash

cd /user/biocore/PACS/CRON/

java –jar filename.jar

#### Config lines added to CRON file

####################

#SYNGOSHARE BIOBANK#

####################

# Run retieveDicom at 5:30, 13:30, 16:30, 20:30 Monday - Saturday

30 5,13,16,20 \* \* 2-6 /user/biocore/CRON/retrieveDicom.sh 1>/user/biocore/CRON/

retrieveDicom.out 2>&1

# Run queryDicom at 5:15, 13:15, 16:15, 20:15 Monday - Saturday

15 5,13,16,20 \* \* 2-6 /user/biocore/CRON/queryDicom.sh 1>/user/biocore/CRON/quer

yDicom.out 2>&1

# Run uploadClinical at 13:45, 21:45 Monday - Saturday

45 13,21 \* \* 2-6 /user/biocore/CRON/uploadClinical.sh 1>/user/biocore/CRON/uploa

dClinical.out 2>&1

# Run queryICEQuestionnaires at 5:00, 13:00, 16:00, 20:00 Monday - Saturday

0 5,13,16,20 \* \* 2-6 /user/biocore/CRON/queryICEQuestionnaires.sh 1>/user/biocor

e/CRON/queryICEQuestionnaires.out 2>&1

### Log/Event system

The Java process’ share a logging class ‘log’. When logging each event is classed into LOG\_LEVELS: *DEBUG*,*INFO*,*ERROR*,*CRITICAL*,*ACTIONREQUIRED.*

If you want to log to the main system, use: log.log();

If the log level is CRITICAL or ERROR, it also gets written to the logfile.

If the log level is CRITIAL or ACTIONREQUIRED, then it also generates an email alert to *CTSU\_CONTACT\_EMAIL.*

### Possible update code requests

\*IMPORTANT NOTE\*

To run the code locally for testing, you will need to remember to set the database.java LINUX = false. Then do remember to change it back before exporting the new JAR. Try and remember to run it locally before uploading to check it fails to connect to the database.

1. If Steve gives you an email address for which to send out reports of missing dicom in studies and incomplete Questionnaire warnings, update BIOBANK\_CLINIC\_EMAIL in logs.java to their specified email address. Compile/Export and upload a new version of uploadClinical and retrieveDicom.
2. If you get told of a keyword to trigger processing of incomplete clinical data. You will need to update the value stored in IGNORE\_INCOMPLETE\_ICE\_DATA with the syngo.share PK of that key. Remember this works on STUDY LEVEL only when adding the tag. Locate the below comment in the file uploadClinical.java.

//kwi is holding the keyword information for this study.

//to find the text string to the keywordPK, use kwi[i2].setKeywordPath() and step

//though till you come across the one you are looking for. Of course, tagging with

//the keyword a study within BIOBANK\_PSI is the first step, so you can find it.

### Tag a study without a PDF uploaded, so it will get processed. Run uploadClinical and step through the code and get the key PK. Update the IGNORE\_INCOMPLETE\_ICE\_DATA value with that, rebuilt, export and upload.

1. If you get told of a keyword to trigger the processing of an incomplete image count in a SERIES, in the same way as the one above, you need to find the syngo.share PK of that key. Easiest way is probably use the same method as above, tag a STUDY with that keyword in BIOBANK\_PSI. Then update *IGNORE\_SERIES\_COUNT\_KEYWORD\_PK in retrieveDicom.java. Recompile, Export and replace the one on the server.* Tagging a keyword on some random data in BIOBANK\_PSI is fine, as they are only use for admin and aren’t downloaded, or processed by us.

### #1 QUERY ICE QUESTIONNAIRES

#### Overview

Monitors the ICE database (the gobz table in the ace\_ice\_live database) for newly-submitted BioBank questionnaires, and fills the biobankparticipant table (in the extract\_pacs\_live database) with key information so that later on, the identities of these DICOM scans can be confirmed and the clinical information can be retrieved if required.

This process is the only one that writes to the biobankparticipant table; other processes merely read from it.

#### Method

The CRON monitors the ace\_ice\_live database for any new entries which aren’t already in the biobankparticipant table. The gtyp is it looking for is 61 (which refers to the participant being welcomed/registered at the centre). It takes the following information from the gobz table and saves it into the biobankparticipant table:

* cid – the table pk, not sure why I’m storing this now.
* Trid and tsig - combines with tsig to give a UID for this entry. To keeps a degree of separation in the data between the ID of DICOM and ID of BIOBANK use these two instead of storing pid and the DICOM patient ID together in any table. If you must at all.
* **pid** - the BioBank participant ID (used in the rest of the Biobank system)
* dicom first/last name - from xub\_isr which is a ROSETTA
* dicom patientid - from xub\_isr which is a ROSETTA

The new entries have their corresponding ROSETTAs decoded and various details are extracted and placed inserted into the biobankparticipant table.

#### Technical details

* Filename: queryICEQuestionnaires.jar
* The program is installed on server: bb5, username: biocore
* At location: /PACS/CRON/queryICEQuestionnaires.jar
* CRON script: /CRON/queryICEQuestionnaires.sh
* CVS location: jonathanpr/eclipse/PACSClient
* Main file: PACSClient\src\uk\ac\ox\ctsu\syngo\client\ queryICEQuestionnaires.java

### #2 QUERY DICOM

#### Overview

This Java CRON job keeps track of DICOM being moved into Authority units which the other Java CRON jobs are interesting in. It stores all relevant information in the table PACSSTUDY, which all other CRON jobs work off. As the table suggests this task works at the study level of DICOM.

#### Method

On each run QueryDicom queries the Syngo.share for all the ‘new’ studies on the PACS in any of the following Authority Units:

* BIOBANK – Where DICOM is downloaded from.
* BIOBANK\_PSI – Where we upload clinical PDFs.
* 1KSCANS – Where the first 1000k scans get copied and treat like BIOBANK\_PSI.

A 'new’ study is currently defined as ALL studies between now and 2 weeks previous to the last time it was scanned. So long as these studies have both a unique StudyPK and AU, it is accepted as a new study in the system. This two week overlap when looking for new studies can be reduced from two weeks once the system is running; should it take up too many resources. Before it adds new information to its table, it tries to match the participant id and/or first/last name into the biobankparticipant table. Each participant on entering a clinic is assigned a randomly assigned 8 character string for both patientid and firstname or lastname. If we get a match on either, then it is entered as confirmedpatientid, and now the identity of this study has been confirmed, it is then available to be processed by other Java CRON jobs.

The last time this CRON job was successfully run is kept in the table DICOMSTUDYDOWNLOADER.

#### Technical details

* Filename: queryDicom.jar
* The CGI is installed on server: bb5, username: biocore
* At location: /PACS/CRON/queryDicom.jar
* CRON script: /CRON/queryDicom.sh
* CVS location: jonathanp/eclipse/PACSClient
* Main file: PACSClient\src\uk\ac\ox\ctsu\syngo\client\queryDicom.java
* It downloads the DICOM files to: download/

### #3 RETRIEVE DICOM

#### Overview

This Java CRON task is to download DICOM series from syngo.share. Once downloaded it generates a manifest for each series, and compresses the series data and manifest into a single zip. The resulting data is submitted into the BioBank storage system for storing.

#### Method

All rows in pacsstudy with au = BIOBANK and aren’t complete (status =1) are retrieved. Each is checked to see if they have a confirmedpatientid set. If not, it tries to match the patiantid or first/lastname into the biobankparticipant table.

Each series contains a data field containing a description of what this series is, each description is checked to see if it has the string “\_count:X” appended. If it does we need to check the number X against the number of images in this series. If we don’t have a full image count, we need to email a report to BioBank, and leave the series for another time.

For all confirmed studies it downloads each series and its attached series keywords. It stores various details of the DICOM file into a manifest CSV file, one for each series, this includes:

* filename – name of the dicom file
* series description – description of the series it is in, taken from the DICOM
* patientid – The confirmed patientid
* studyid – DICOM study id
* study date – Date this study was inserted into the PACS
* modality – modality
* aet – AET of the sender
* hostname – Hostname of the sender
* uncompressed size (in bytes)) – size of the file in bytes

It then compresses each DICOM file in each series along with the manifest file and adds the details to the dicombiobank table.

It then adds an entry into the import table of the biobank system, notifying the BioBank system that there is data ready to be archived. Once accepted into the system and the archiving is complete, the BioBank system removes the original file for us.

Filename of the resulting zipped DICOM series has the following format: modality\_trid\_cid\_studyPKseriesPK.zip

#### Technical details

* Filename: retrieveDicom.jar
* The CGI is installed on server: bb5, username: biocore
* At location: /PACS/CRON/ retrieveDicom.jar
* CRON script: /CRON/ retrieveDicom.sh
* CVS location: jonathanp/eclipse/PACSClient
* Main file: PACSClient\src\uk\ac\ox\ctsu\syngo\client\ retrieveDicom.java

### #4 UPLOAD CLINICAL

#### Overview

This is the Java process which is responsible for collecting the ICE questionnaire for all the scans in the AU BIOBANK\_PSI, from which it create the PDF and upload it to the study in question.

#### Method

It retrieves all unprocessed rows from the table pacsstudy; these have the au = ‘BIOBANK\_PSI’ OR ‘1KSCANS’ and status 1. If the row doesn’t have a confirmedpatientid, it tries to match the patientid or first/lastname into the table BIOBANKPARTICIPANT dicomparticipantid or firstnameid. If we get a match, it uses the corresponding pid as an index and extract from ICE datasbe the Biobank questionnaire’s required clinical information, from which we generates and uploads to the PACS a PDF. The participants’ answers are retrieved from ace\_ice\_live and the corresponding question text is retrieved from ace\_trove\_live. See Appendex B for more information on how this is done.

Before it can request data from ICE, it needs to know all the relevant information has been processed and uploaded into ICE. As a participant visits each station in the clinic, we get an entry in the gobz tables of ice\_live. Each gobz has a gtyp identification number which corresponds to an event in the clinic. The information that we require has the gtyps of 11,22,31,34. If any of them are missing, we will need to look for the presents of gtyp 51, which indicates the participant is fully processed. If gtyp 51 is present, then we extract what is there and generate as much clinical information as we can.

If after 24 hours we still don’t have a full set of gtyps and we are still missing gtyp 51, we sent off an alert email so this can be investigated. The system will only send out one email per day/per study. On receipt of the email they can resolve this in various ways:

1. Fix the issue on their side by performing tasks which add these missing data rows.
2. By tagging the series in question with a keyword (currently this is undefined).

See Appendix B for further details on the ace database.

Once generated the PDF is uploaded to the corresponding au of the scan. The PDF will contain any of the following it can from the ICE data:

* Age
* Sex
* Is a smoker
* past smoker
* systolicBP
* disstolicBP
* Height
* Weight
* Alcohol consumption
* Employment Status
* Ethnic Group
* as well as listing self-reported:
  + Illnesses and Cancers
  + Non caners
  + Operations

#### Technical details

* Filename: uploadClinical.jar
* The CGI is installed on server: bb5, username: biocore
* At location: /PACS/CRON/ uploadClinical.jar
* CRON script: /CRON/ uploadClinical.sh
* CVS location: jonathanp/eclipse/PACSClient
* Main file: PACSClient\src\uk\ac\ox\ctsu\syngo\client\ uploadClinical.java

### #5 DECRYPT DATABASE

#### Overview

Decryption of all encrypted fields in a table.

#### Method

Scans the assigned table and looks for where statusenc = 1, for each it decrypts the encrypted columns. It does this by extracting all columns which ends with \_enc, decrypts those strings and inserts the decrypted strings into the corresponding tables columns that don’t end with \_enc. It uses AES encryption which comes bundled with Java.

Note: Each table which has encryption contains the column aesvi which is the Initialization vector used by the AES for that row.

#### Technical details

* Filename: decryptdb.jar
* The CGI is installed on server: bb5, username: biocore
* At location: /PACS/CRON/decryptdb.jar
* CRON script: /CRON/decryptdb.sh
* CVS location: jonathanp/eclipse/PACSClient
* Main file: PACSClient\src\uk\ac\ox\ctsu\syngo\client\ cmdLineDecryptDB.java

## 

## WEB SITE

#### Overview

We provide a basic web interface; this interface provides an overview of the CRONs status, DICOM processing status and a way to manually match up scans to their ICE participant’s id. It restricts access to the site using the ICE sign in system; it also checks the table ‘stuff’, a table which holds usernames in the ICE system which are allowed to access this web site. There is also a corresponding CGI which the JavaScript calls.

#### Technical details

*Web page*

* The web site is a C++ program, running as a CGI.
* It is called: pacswebsite.cgi
* The CGI is installed on server: biocore@bb5
* At location: \*UNDEFINED\*
* CVS location: jonathanp/BioBankPACStatusCGI
* Main file: BioBankPACStatusCGI/main.cpp

*CGI which the javascript calls to interact with the database*

* The CGI app is written in C++.
* It is called: pacswebsitecoms.cgi
* The CGI is installed on server: biocore@bb5
* At location: \*UNDEFINED\*
* CVS location: jonathanp/BioBankPSICGI
* Main file: BioBankPSICGI/main.cpp

# APPENDIX A

### Definitions

**AU**

Authority Unit. This is an area, or ‘partition’, on the Syngo.share PACS. Each one needs authorisation to view or modify the containing data, so only those who should, can access the medical data.

**ICE**

a database (ace\_ice\_live) containing anonymised data pertaining to participants visiting the Cheadle clinic

e.g. data from questionnaires given to participants; records that a certain patient had a certain scan at a certain time

**PSI**

Potentially Serious Incident. Refers to a possible medical issue noticed by a radiographer, radiologist or other specialist (e.g. a lump is spotted on an image that has an appearance similar to cancerous lumps)

**PACS**

Picture Archiving and Communication System. This refers to systems that abide by the PACS standards governing how such systems should operate.

VNA - Vendor Neutral Archive, basically another name for a PACS.

Syngo.share - The VNA software supplied by Siemens to manage the PACS.

MRI – Magnetic Resonance Imaging

Modality – Refers to a piece of imaging equipment. Each DICOM file stores what type of medical modality it is, these are two letters. See Appendix F for a list of two letter modalities and their description.

DICOM – In this document this refers to the data format part of the DICOM standard. Generally speaking a single DICOM file contains a single image (or slice) of a much larger set, see Study and Series.

(DICOM) Study and Series – A study is a grouping a DICOM files which would normally\* come from a single visit to a device. The more advanced devices can produce various types of data, or perform repeated scans/snap shots over time; these are normally stored as separate series within a study.

\*normal practice is what you would see most of the time, there is always the odd rare exception.

# APPENDIX B

### The clinical information from ICE\_LIVE

Within the database ACE\_ICE\_LIVE, the table gobz stores information from every event at a BioBank clinic. Each event at the clinic is stored as a new row in the gobz table. The column gtyp specifies which event type it refers too.

The table below details the values we are interested in retrieving from the system to build up the clinical information. To find the id number of the question text, you have to enter the trove field id into <http://bb5.ctsu.ox.ac.uk/crystal/qsearch.cgi?sa=1&sd=1>, which gives you the index into the table ace\_trove\_live.

Prefixes used to decode the ROSETTA value:

* o. value to be found in column xub\_osr
* i. value to be found in column xub\_isr

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Gtyp | Value | Trove field |
| Age | 11 | o.dob | calculate from date of visit (tsig on record) |
| Sex | 11 | o.male | T or F |
| List of self-reported cancer and non cancer illnesses + dates (year) of diagnosis | 31 | i.data.Vcanc.count  i.data.Vcanc.c#  i.data.Vcanc.w# | 82  84 |
| List of self-reported operations + dates (year) of diagnosis | 31 | i.data.Vnonc.count  i.data.Vnonc.c#  i.data.Vnonc.w# | 85  87 |
| *Current employment status* | 21 | i.answer.a.D9.ans.ans# | 6142 ans: [100295](http://bb5.ctsu.ox.ac.uk/crystal/coding.cgi?id=100295&sd=1&sa=1) |
| *Ethnic group* | 22 | i.answer.a.YE1.ans | 1657 |
| *Smoker – ever/never/current* | 22 | i.answer.a.S1.ans  i.answer.a.S4AA.ans | 1239  1249 |
| *Alcohol – previous or current heavy drinker* | 22 | i.answer.a.A1.ans | Aswsers:[100402](http://bb5.ctsu.ox.ac.uk/crystal/coding.cgi?id=100402&sd=1&sa=1). |
| *Systolic BP* | 31 | i.data.bp#.sys  i.data.bp#\_sys | 4080  93 |
| *Diastolic BP* | 31 | i.data.bp#.dia  i.data.bp#\_dia | 4079  94 |
| *Weight* | 34 | i.data.imped.wgt  i.data.weight | 3098  3160 |
| *Height* | 34 | i.data.hstand | 50 |
| *BMI* |  | calculate from weight & height | |
| Dicom Participant ID | 61 | i.iermit.idid | |
| Scan date | 61 | i.tsig | |
| Medication History | 31 | i.data.Vtreat.d# 88  i.data.Vtreat.count | |

**APPENDIX C**

**Java notes**

* biobore@bb6 has Java 1.6 running.
* extract@bb6 has java 1.4 running. Looks like we are now only using biocore user account exclusively so this no longer matters.
* To debug the SSL connection, run the java jar with:
  + Java –Djavax.net.debug=ssl –jar filename.jar
* If you get the error:
  + “java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty”
  + You are most probably missing the following file from the directory the java.jar is: webservice\_api.jks
* Compiling and getting an updated java app live:
  + In Eclipes, do File/Export
  + Choose Java/Runnable JAR file. Next.
  + Change launch configuration to the one you want to generate.
  + Select an export location/Filename on your local machine
  + Set library handling to : Package required libraries into generated JAR
  + Finish generates the jar.
  + Upload the jar to biocore@BB5 /PACS/CRON and overwrite the one you want to replace.

# APPENDIX D

### Syngo.Share PACS

This central PACS running syngo.share is situated at CTSU. To access the syngo.share web interface go to: <https://syngo.ndph.ox.ac.uk/>. If you fail to connect, you are probably blocked by the firewall.

Notes:  
The AUs go under the heading “Producer” for some reason.

Right-click on a study/series and select “Index” to access keywords

# APPENDIX E

### Upgrading the syngo.share library

Notes on upgrading the library interface to a more recent version which is used to interface into syngo.share. The syngo.share commination layer is based on Apache Axis 2 and uses rampart for its security, more information on this from <http://axis.apache.org/axis2/java/core/>, http://axis.apache.org/axis2/java/rampart/

1. Download the end point of the service from the web admin section of the syngo.share. You will need to sign in with an administrator account to access this. Go to Advanced administration tools -> webservices. This page lists the services and their corresponding https addresses from which you can access them with from the API.

On time of writing there are two versions available, VA15 and VA20. The higher the number, the more recent is the version. I recently upgraded to running with VA20.

Using the menu Webservice resources, download both Servercertificate (jks) and Technical API description (WSDL). You will be asked for a password on downloading the certificate, do remember this as you will need to use it in your code to open the certificate.

1. Download and extract Axis2 standard distribution.
2. Generate the java subs from the WSDL by:

* Extract the files from the WSDL.
* Locate wsd12java (in the axis2 bin directory)
* For each file in the WSDL run:

Wsdl2java –url filename.wsdl

This will generate the code stubs, which you import into your Java project.

1. It goes without saying, you will need an account on syngo.share for your API calls to access the system with, make sure it has enough privileges to perform the functions you require. When calling the code in the subs you generated above, you will need both the URL for the called service, the Servercertificate, plus password from part 1. The Axis2 web documentation suggests enabling MTOM for faster data transfer. The rest should be straight forward.

# APPENDIX F

### Modality abbreviations

|  |  |
| --- | --- |
| Modality | Description |
| BI | Modality of type Biomagnetic Imaging |
| CR | Modality of type Computed Radiography |
| CT | Modality of type Computed Tomography |
| DG | Modality of type Diaphanography |
| DX | Modality of type Digital Radiography |
| ECG | Modality of type Electrocardiograms |
| EM | Modality of type Electron Microscope |
| ES | Modality of type Endoscopy |
| GM | Modality of type General Microscopy |
| HC | Modality of type Hard Copy |
| IO | Modality of type Intra-oral Radiography |
| LS | Modality of type Laser Surface Scan |
| MG | Modality of type Mammography |
| MR | Modality of type Magnetic Resonance |
| NM | Modality of type Nuclear Medicine |
| OP | Modality of type Ophthalmic Photography |
| OPM | Modality of type Ophthalmic Mapping |
| OPR | Modality of type Ophthalmic Refraction |
| OPV | Modality of type Ophthalmic Visual Field |
| OT | Modality of type Other |
| PT | Modality of type Positron Emission Tomography (PET) |
| PX | Modality of type Panoramic X-Ray |
| RD | Modality of type Radiotherapy Dose (a.k.a. RTDOSE) |
| RF | Modality of type Radio Fluoroscopy |
| RG | Modality of type Radiographic Imaging (conventional film screen) |
| RTIMAG | Modality of type Radiotherapy Image |
| RP | Modality of type Radiotherapy Plan (a.k.a. RTPLAN) |
| RS | Modality of type Radiotherapy Structure Set (a.k.a. RTSTRUCT) |
| RT | Modality of type Radiation Therapy |
| SC | Modality of type Secondary Capture |
| SM | Modality of type Slide Microscopy |
| SR | Modality of type Structured Reporting |
| TG | Modality of type Thermography |
| US | Modality of type Ultrasound |
| VL | Modality of type Visible Light |
| XA | Modality of type X-Ray Angiography |
| XC | Modality of type External Camera (Photography) |

# APPENDIX G

## Functional Testing

Testing the following processes:

* queryDicom.jar – requires database and syngo.share access
* queryICEQuestionnaires.jar – requires database access
* retrieveDicom.jar - requires database and syngo.share access
* uploadClinical.jar - requires database and syngo.share access
* web site – requires database access.

## General functionally shared between all java processes

|  |  |  |
| --- | --- | --- |
| Function | Id | Tested in test # |
| Will not run if already running | 1 |  |
| Will alert the best it can if it can’t contact syngo.share or database | 2 |  |
| Will set lastrun times on start and finish running | 3 |  |

## queryDicom functional test table

|  |  |  |
| --- | --- | --- |
| Function | id | Tested in test # |
| Will pick up all new studies not already know about | 4 |  |
| Will ignore studies already known about, when it comes across them | 5 |  |
| Will match either name or patientid into biobankparticipant where possible | 6 |  |
| Will correctly save all required study information into pacsstudy | 7 |  |
| Will update lastdownload on successful run | 8 |  |

## queryICEQuestionnaires functional test table

|  |  |  |
| --- | --- | --- |
| Function | Id | Tested in test # |
| Will pick up all new ICE participants not already know about | 9 |  |
| Will ignore studies already known about, when it comes across them | 10 |  |
| Will save following data from ICE.GOBZ into biobankparticipant:  cid,trid,pid,tsig and dicom name,pid from xub\_isr ROSETTA | 11 |  |

## retrieveDicom functional test table

|  |  |  |
| --- | --- | --- |
| Function | Id | Tested in test # |
| If a series identity is not confirmed it was try and match up as in #6 | 12 |  |
| If a series can’t be matched and is over 24 hours old, it emails an alert. It will only send out 1 alert/study/day. | 13 |  |
| If a series is not confirmed after attempted match, it is left unprocessed | 14 |  |
| It fails to download every series in a study, it will try the study again next time, but it will only download each series one. | 15 |  |
| Before downloading a series, it will check the description for a image count. If the series has less images, and doesn’t have a keyword saying this is OK, it will report this and skip the series. | 16 |  |
| It saves all keywords attached to the series (ignores any study level keywords). Into keyword and dicombiobank\_keyword | 17 |  |
| For each dicom file in a series a CSV line in a manifest is created. | 18 |  |
| Each DICOM file is zipped up by series, each series will have its own manifest file. The file name will have the following format:  Modality\_tridcid\_StudyPK\_Series\_PK.zip | 19 |  |
| The zip will be located \*unknown\* for archiving | 20 |  |
| A line for each zip will be added to the BioBank import table | 21 |  |
| The zips will be accepted into the BioBank system | 22 |  |
| After the DICOM is zipped, the original files will be removed off the system | 23 |  |

## uploadClinical functional test table

|  |  |  |
| --- | --- | --- |
| Function | Id | Tested in test # |
| It will only pickup on studies which have not already had their clinical information uploaded | 24 |  |
| If a study is not confirmed it was try and match up as in #5 | 25 |  |
| If a study can’t be matched and is over 24 hours old, it emails an alert. It will only send out 1 alert/study/day. | 26 |  |
| If a study is not confirmed after attempted match, it is left unprocessed | 27 |  |
| Will generate a clinical PDF with the file name: PatientID\_studyPK.pdf | 28 |  |
| Will upload the clinical PDF only to BIOBANK\_PSI | 29 |  |
| Unless it has a keyword attached saying to process it, it will leave any studies without complete the ICE questionnaire result available. No gtyp 51. | 30 |  |
| It will pick up the previously specified data from ICE when it is available and add it to the PDF. See ‘THE JAVA CRONS AND THEIR PROCESS DESCRIPTIONS’. | 31 |  |
| The PDF will have a second page showing the DICOM ID and an entry box to write a report. | 32 |  |
| Once uploaded the local PDF will be deleted. | 33 |  |

## web site functional test table

|  |  |  |
| --- | --- | --- |
| Function | Id | Tested in test # |
| Sign in username has to be in Staff table to sign in | 34 |  |
| Both username and password are verified against the ICE sign in system to get access. | 35 |  |
| It displays all failed matches | 36 |  |
| A failed match can be corrected, it is not show in future views of the page, and is picked up by Java processes and processed | 37 |  |
| A failed match can be ignored, it is not show in future views of the page, and is ignored by all Java processes. | 38 |  |
| It displays the correct status information | 39 |  |

## TEST MATRIX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Test 6 | Test 7 |
| 1 | X |  | X | X | X | X | X |
| 2 | X |  | X | X | X | X | X |
| 3 | X |  | X | X | X | X | X |
| 4 |  |  |  | X |  |  |  |
| 5 |  |  |  | X |  |  |  |
| 6 |  |  |  | X |  |  |  |
| 7 |  |  |  | X |  |  |  |
| 8 |  |  |  | X |  |  |  |
| 9 |  |  |  | X |  |  |  |
| 10 |  |  |  | X |  |  |  |
| 11 |  |  |  | X |  |  |  |
| 12 |  |  | X |  | X |  |  |
| 13 |  |  | X |  | X |  |  |
| 14 |  | X |  |  | X |  |  |
| 15 |  |  |  |  | X |  |  |
| 16 |  |  |  |  | X |  |  |
| 17 |  |  |  |  | X |  |  |
| 18 |  |  |  |  | X |  |  |
| 19 |  |  |  |  | X |  |  |
| 20 |  |  |  |  | X |  |  |
| 21 |  |  |  |  | X |  |  |
| 22 |  |  |  |  | X |  |  |
| 23 |  |  |  |  | X |  |  |
| 24 |  |  |  |  |  | X |  |
| 25 |  |  |  |  |  | X |  |
| 26 |  |  |  |  |  | X |  |
| 27 |  |  |  |  |  | X |  |
| 28 |  |  |  |  |  | X |  |
| 29 |  |  |  |  |  | X |  |
| 30 |  |  |  |  |  | X |  |
| 31 |  |  |  |  |  | X |  |
| 32 |  |  |  |  |  | X |  |
| 33 |  |  |  |  |  | X |  |
| 34 | X |  |  |  |  |  | X |
| 35 | X |  |  |  |  |  | X |
| 36 |  |  |  |  | X | X |  |
| 37 |  |  |  |  | X | X |  |
| 38 |  |  |  |  |  |  | X |
| 39 | X |  |  |  | X | X |  |

## The tests

### Initial state before each test

Clear down all databases excluding cronstatus,staff

Clear up all logs/ dicom files already in the system

Create/empty 3 AU

BIOBANK\_TESTING

BIOBANK\_PSI\_TESTING

FIRST\_1K\_TESTING

The java code is updated to use these AUs.

## Test 1 - Initial Error testing

1. Turn off the networking. Run all Java Apps. Check Results.
2. Turn back on networking, change the syngo.share api account password. Run all Java App.
3. Run each java process in turn from the IDE, leaving a break point so they don’t complete. Run the same one again from the command line while the break point is in place.

### Expected Results

1. No Syngo.share correct and no Database connection. It can’t email out for help, or write anything to the database, it will write an error to the log file. Should be 4 entries.
2. With no Syngo.share access it will write an error into the database. It should also send out an email as the sign in failed. No other changes are expected to anything part from the log entries in the database log table, and log file. There is 1 no syngo.share access errors in the database logs from queryDicom java process. The others which use Syngo.share won’t have any reason to access it yet, so won’t be complaining.
3. The main web site will show 1 error reports in the log page.
4. On running the command line java process, each should shut down and not do anything.

## Test 2 - Initial state testing

1. Run all java processes, apart from QueryICE.

### Expected Results

As there is no data in the system, there should be no state change.

## Test 3 – No Match testing

1. Move to BIOBANK\_TESTING two studies.
2. With no data in Biobankparticipant table, there won’t be any matches. Change one study date in syngo.share to today, and the other 1 week old.
3. Run queryDicom
4. Run retrieveDicom
5. Run retrieveDicom

### Expected results:

When retrieveDicom is run; for the study which is older than 24 hours, an alert email is sent. Check from the email details that the study can located on singo.share. Nothing else should happen; there should only be a single email. A check on the web site can confirm the current status.

## Test 4 – queryDicom and queryICEQuestionaires testing

1. Run queryICEQuestionnaires
2. Move to BIOBANK\_TESTING a study with a match in biobankparticipant
3. Run queryDicom
4. Move to BIOBANK\_TESTING a study with no match in biobankparticipant
5. Change the lastname of the study in syngo.share to match another studies firstname in biobankparticipant
6. Run queryDicom
7. Move to BIOBANK\_TESTING another study with no match in biobankparticipantpant .
8. Run queryDicom
9. run queryICEQuestionnaires

### Expected results:

We should have confirmed matches for #2 and #4. No match for #7. There should be a single entry in pacstudy for both which match. All the information in that table should be check and should be accurate. Last run time in dicomdownloader should be updated.

Biobankparticipant should not have any cid, trid duplications. Value saved into this table should be checked for accuracy.

Two entries in cronstatus should also have been updated.

## Test 5 – retrieveDicom testing

Checks should be made on the web site to check the status are correct.

1. Move to BIOBANK\_TESTING: 2 studies, one which will match and 1 which won’t.
2. Add various QA keyword to the series with in the studies.
3. Change a series description in the match study (using singo.share) and append ‘\_count:434’
4. Run queryDicom
5. Run retrieveDicom
6. Run queryICEQuestionnaires
7. Run retrieveDicom
8. Use the keyword tag on the series with the incorrect \_count: appended, so it will be processed.
9. Run retrieveDicom
10. On the web site, ‘correct’ the unmatched study identity, it doesn’t matter to what.
11. Run retrieveDicom
12. Check at a later point that the zips have been removed by the biobank archive system and the data has been archived.

### Expected results:

Part 1-5: Two emails are set out as the first time queryICEQuestionnares is run, there is no match on biobankparticipant table and the study dates are older than 24 hours. The series are left unprocessed and there are no entries in dicombiobank.

Part 6-7: The matching Study will have all its series downloaded (along with keywords) apart from the one with the \_count434: appended.

Part 7-8: With the keyword added to the series, this final series will be downloaded along with the keywords.

Part 9-10: The newly matched study will have all its series and keywords downloaded.

Check against the syngo.share we have all keywords with the correct series. All dicom data downloaded. The corrected patient id’s are now in place and are used for the corrected participant.

The series keywords will be saved into dicombiobank\_keyword, the definitions of the keywords will be entered into keywords. The each dicom file will be downloaded and zipped, with a generated manifest added. All data will be saved in the ‘correct’ location for archiving.

An entry will be written into BioBank import table for every zip.

## Test 6 – uploadClinical testing

Checks should be made on the web site to check the status are correct.

1. Move to BIOBANK\_PSI\_TESTING: 2 studies, one which will match and 1 which won’t. The one which matches should have full clinical data available.
2. Run queryDicom
3. Run uploadClinical
4. Run queryICEQuestionnaires
5. Run uploadClinical
6. On the web site, ‘correct’ the unmatched study identity, change to an identity with no clinical information, or not a full set.
7. Run uploadClinical
8. Tag the series with the ‘process clinical anyway’
9. Run uploadClinical

### Expected results:

Part 1-3. Two emails are set out as the first time uploadClinical is run, there is no match on biobankparticipant table and the study dates are older than 24 hours. The series are left unprocessed.

Part 4-5. A study is matched. Data is extracted from ICE, a clinical PDF is generated and uploaded. The status of pacsstudy for that study is updated to complete.

Part 6-7. An email is set out, if the study is over 24 hours old, stating that we have incomplete ICE data.

Part 8-9. A pdf is generated with what information (if any) is available, and uploaded to the study.

A single PDF will be loaded to each study. The PDF filename will be of the correct format: PatientID\_studyPK.pdf. No PDFs will be left on the server after being uploaded.

## Test 7 – web site testing

1. Sign on to web site with a random username string and password
2. Add a fake user account name to staff and sign on with a random password
3. Sign on with a valid CTSU user account, which isn’t in staff table.
4. Sign on with a valid username, password which is in staff table.
5. Add a Dicom study to BIOBANK\_TEST which will get a match on ICE
6. Run queryICEQuestionnaires
7. Run queryDicom
8. Use the web site and set the system to ignore the study.
9. Run retrieveDicom

### Expected results:

1. Failed to sign in
2. Failed to sign in
3. Failed to sign in
4. Successfully sign in

5-9 The study is not downloaded and is ignored.

# APPENDIX H

## A few possible problems and their solutions

### Unable to match a study to its BioBank ID, someone typed it in wrong, the web site is broken so we can’t manually match the study.

1. Find the study entry in table pacsstudy. Update the row with the correct participant id in confirmedpatientid. It will then be picked up by the other Java processes.

### There is a study which is generating problems, how do I tell the system to ignore it. This may happen if say the DICOM is deleted off the PACS before it’s processed (This will never happen in real life, of course). It also may be the case that you are getting alert messages from test data which will never match.

1. If it’s a non-confirmed study, then you can choose to ignore it using the web site.
2. If it has been confirmed, or the web site isn’t work, find the study in pacsstudy and set the status to 3.

### I’m getting alert messages from test data which will never match or I just want it to STOP!

1. If it’s a non-confirmed study, then you can choose to ignore it using the web site.
2. If it has been confirmed, or the web site isn’t work, find the study in pacsstudy and set the status to 3.

### They are now have an AU ‘1kScans?’, they want Clinical PDFs to be uploaded just like BIOBANK\_PSI. How do I make this happen?

In queryDicom.java, update the line (top of class):

**private** **static** String[] *AUList* = {"BIOBANK","BIOBANK\_PSI"};//,"1K\_SCANS"};

to include the new AU. Do replace 1K\_SCANS with the final name of it.

In uploadClinical.java, update the line (top of class):

String[] AUList = {"BIOBANK\_PSI"};//,"1K\_SCANS"};

Recompile, export and replace the jar files on BB5.

### They have a PSI and we have not uploaded the clinical info, they want it uploaded NOW! How do I do it manually?

1. Find the row which matching patient id/name in biobankparticipant. The pid is the pid (unique key) for the participant in the ACE\_ICE\_LIVE, which holds all the information/data on events while at the clinic.
2. If there is no entry, then either there is no data in ICE on this ID, or queryICEQuestionnaires isn’t working, or perhaps it very new and hasn’t been pickup yet. Running queryICEQuestionnaires.sh in the CRON directory of BB5, will trigger it to fill the biobankparticipants. If this process isn’t working for some reason, then the only option is to step though the program and find out why, as the pid is stored in a ROSETTA so there is no other way of getting it without extracting all the data till you find it, and this process is really the only thing which queryICEQuestionnaires performs.
3. Once you have the pid, then appendix B is the information I got on how to extract the questions and answers from the database ICE\_LIVE. To use some code which can do this for you, create a java class and add:

**public** **static** **void** main(String[] args)

{

//setting the pacstudy information sets those values in some of the text output.

db\_pacsstudy sr;

sr.setConfirmedPatientID(“ABC”); //Change to patientID of the DICOM

sr.setFirstName(“123”); //Change to the first or last name of the DICOM

questionaire q1 = **new** questionaire(“213”); //Change to the Biobank’s pid

Vector<pair<Integer,String>> quizVector = q1.getReportStrings(sr);

//Iterate though each pair, each entry is a line:

// Integer - is the indentation used for that line

// String – is the text.

// Now dump this out with system.out.println() and cut and paste it into word or something and generate a PDF.

//use singo.share to upload it against the scan in BIOBANK\_PSI

}

The guys at Biobank want alert emails when we don’t have full ICE data or an we have an incomplete DICOM series. How to I make them happy?

Logs.java, update BIOBANK\_CLINICAL\_EMAIL with their email.

Unfortunately, you then need to Export and replace retreieveDicom.jar and uploadClinical.jar

### Stuff isn’t working. What should I do? Where should I start looking?

1. Look at the web site’s log page, any issues being logged? Alternatively look at the logs table in biobank\_pacs\_live.
2. Check the status page, when did they last run? (or table cronstatus). Check against the times in this document on when they should have last run. Is the CRON firing them off OK? If not try running each of the javaprocessname.sh in Biocore@BB5/CRON, did they run OK? Or perhaps they have been running for hours or days (finish time is older then start time). Is there a lock on any of the databases? The only job which could take hours is the dicomdownloader, all others should be ‘much quicker’.
3. In the directory on the bb5 where the java apps live (PACS/CRON directory), there is a log file in the same directory (logfile.txt), if all else isn’t working, it should at least be writing something to this if it’s running at all, even if it can’t access db or email for help.
4. Can you turn BB5 on and off again?
5. Get the thumb screws out and persuade a DBA to accept all blame (steal their lunch money while you are at it).

### Stuff isn’t working, TURN IT OFF!

1. Sign on to Biocore@bb5.
2. Cd /PACS/CRONS
3. rm \*.jar