

# Behavior Analysis of an Infrastructure for Access Control Secure Medical Image Sharing between PACS and DI-r

## Introduction

This document describes various components in the architecture, especially various repositories and registries, what they contain and their functionalities. Further, we describe the simulation in IBM Rational Rhapsody using a case study.

## Architecture

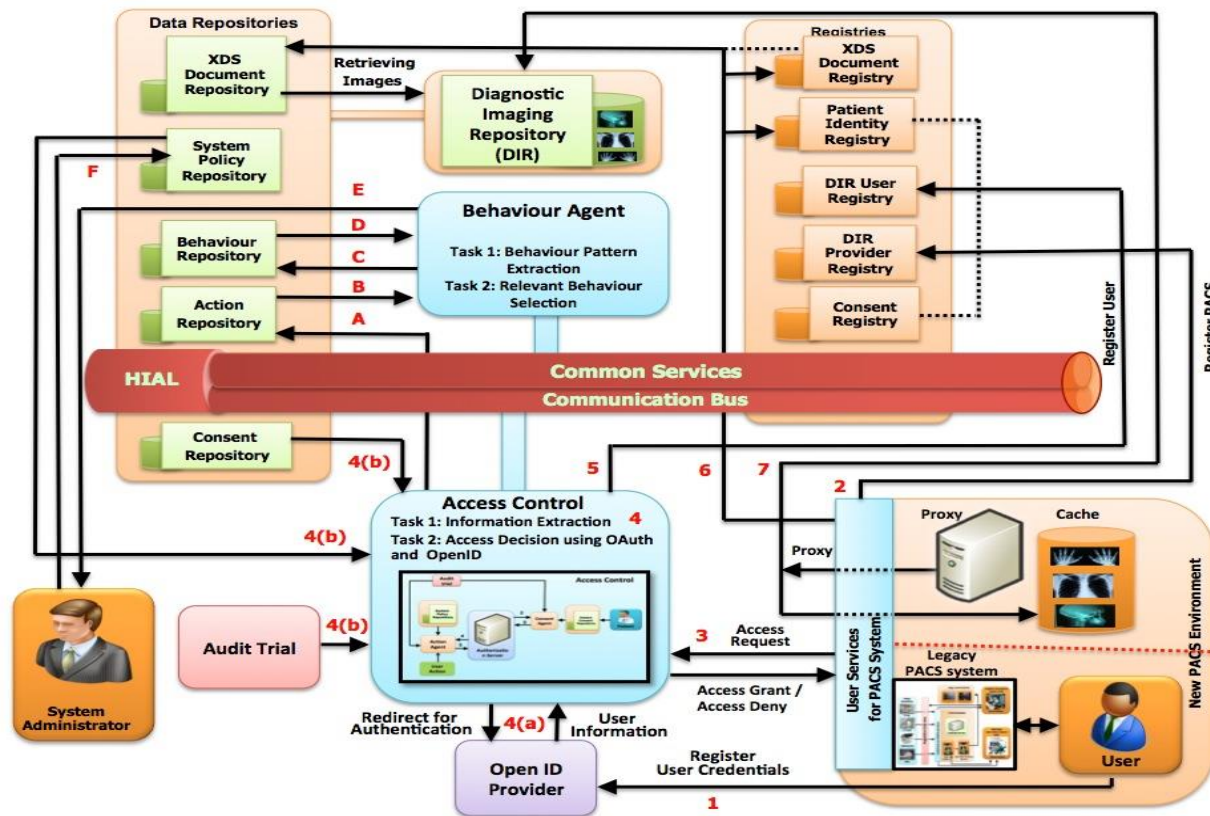


Fig 1: Proposed Infrastructure

## Different Registries, Repositories and their contents:

Repositories and registries are employed for documentation, maintenance and preservation of patient and user details. We make use of all these repositories and registries to anonymize user details. By doing so we make sure that, the medical image and the image details are separated and stored in different locations. They are linked within the system by generating appropriate IDs. Hence, unauthorised users cannot

access images directly by intruding into the system. Following are the ones that we have employed in our system.

### **Patient Identity Registry (PIR)**

Medical Images when captured by image modalities will contain meta data associated with the image. Normally this metadata will contain patient details such as name, sex, date of birth, Health card number, age, address, contact number, etc. Prior to storing Images in DIR, we register patient with a global registry called the Patient Identity registry. The purpose of assigning this ID is to assign them a master ID irrespective of the local ID that they already have from the home location.

### **XDS Document Registry**

The purpose of this registry is to store the XDS ID of patients. Image metadata is stored in XDS Document repository and is identified by using patient XDS ID. This ID is linked to patient ID in PIR. As we already discussed, ID in PIR is linked to patient details. In short, the patient details and image metadata are stored separately within the system.

### **DI-r provider Registry**

Diagnostic Imaging repository stores the medical images and associated meta data from different vendors that are located in different locations. For the PACS to store their images in DI-r, they have to register themselves with DIR provider registry as image providers. Upon doing so, each provider will be assigned a DIR provider ID. Purpose of registering providers is to make sure that only authorised image providers (PACS) and the associated users can approach the system for image storage and retrieval.

### **DIR User Registry**

User who approaches the system for image retrieval is undergone through various checks prior to granting access token. Once the user has successfully cleared the checks he is registered with the DIR User Registry. At a later stage when the user approaches the XDS Document registry with the token, user ID in the DIR user registry is checked to make sure that he has proper access rights.

### **System Policy Repository**

System policy Repository has the system security policies defined by the administrator. These policies restrict various roles in the hospital from accessing the patient details. These roles include physician, nurse, radiologist, lab technician, etc. Policy is defined for

different roles. Permission is granted by considering the user location, type of image, purpose of access, time and date of access, etc. Parameters can vary depending on the requirement of the system.

### **Consent Repository**

Consent repository has consents defined by the patient. Access is granted only if the patient consent is allowing the user to access. Consent is defined by a patient for a particular user includes parameters such as the name of a user, user role, his location, type of image, time and date the user is allowing, purpose of retrieval etc.

### **Action Repository**

Action repository stores the user action every time the user approaches the system. Action consists of user information that is the user first name, last name, role, location. In addition to it the data in the access request user ID, patient first name, patient last name, health card no, dob, type of image, purpose of retrieve, access request time and date. This is similar to an audit trail, which is a record of access requests made by different users at different time.

### **Behavior Repository**

User Behavior is extracted from the user action stored in the action repository. We analyze user action for a period and extract frequently occurring pattern. For example, a physician accessed the MRI image of a patient consequently for 5 days a week almost same time every day. We look into the action repository and count the number of times each of the patterns has occurred. The most frequently occurring patterns are extracted and we call it the behavior of the user and store it in behavior repository.

### **XDS Document Repository**

This repository has all the meta data associated with the image. It includes patient XDS-ID, type of image and its availability, captured location, date, and Di-r image no. When a user approaches the XDS Document registry, the token he submits will have a reference to this registry, thereby providing the metadata associated with the image and the index associated with that particular number.

### **Diagnostic Imaging Repository(DI-r)**

Di-r is the database where all the medical images are stored. Images are stored with the associated XDS ID. All other details related to the image are stored in different repositories.

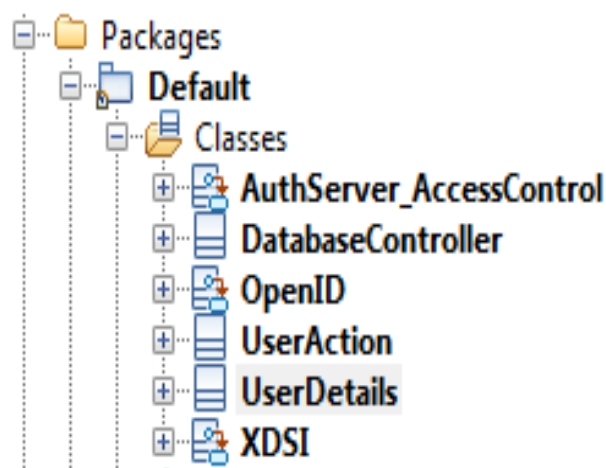
## Case Study: Simulation results using IBM Rational Rhapsody and MySQL DB

As you can see in Fig 1, system does three main functionalities:

- OpenID Authentication
- OAuth based Access Control
- XDS-I profiles based Image retrieval

The following case study will explain the process of image retrieval by using the methodology proposed in the architecture.

### Different classes in IBM Rational Rhapsody



**Fig 2: Classes in Rhapsody**

In Fig 2, we see different classes employed in Rhapsody. The classes: 'OpenID', 'AuthServer\_AccessControl' and 'XDSI' handle the main functionalities of the system. Class 'DatabaseController' contains various operations that connect Rhapsody to MySQL DB for accessing data in DB. Lastly, we use the classes 'UserAction' and 'UserDetails' to handle the attributes involved and carrying out other minor functionalities involved while running simulation. Now we go into the details of functionalities of each of these classes. We can explain the working of the system by using the following scenario :

Dr. Brown Kyle wants to access the image of patient Mr. Matt Basar

#### User Details:

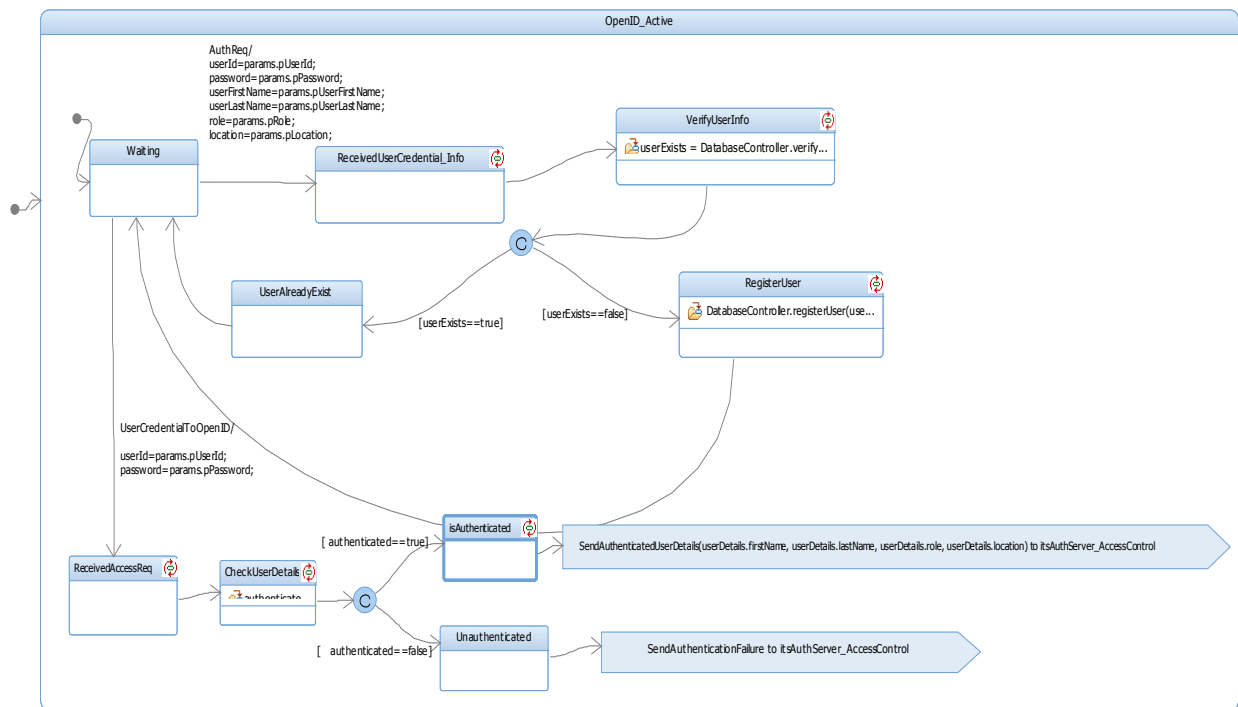
- Brown Kyle
- Role : Physician

- Location : RVHS
- userID : 100466613

### Patient Details:

- Patient Name: Matt Basar
- Health Card No : 100493164
- DOB: 1957-01-02
- Type of image: MRI
- Purpose: Diagnosis

### Step 1 : Authentication with OpenID module



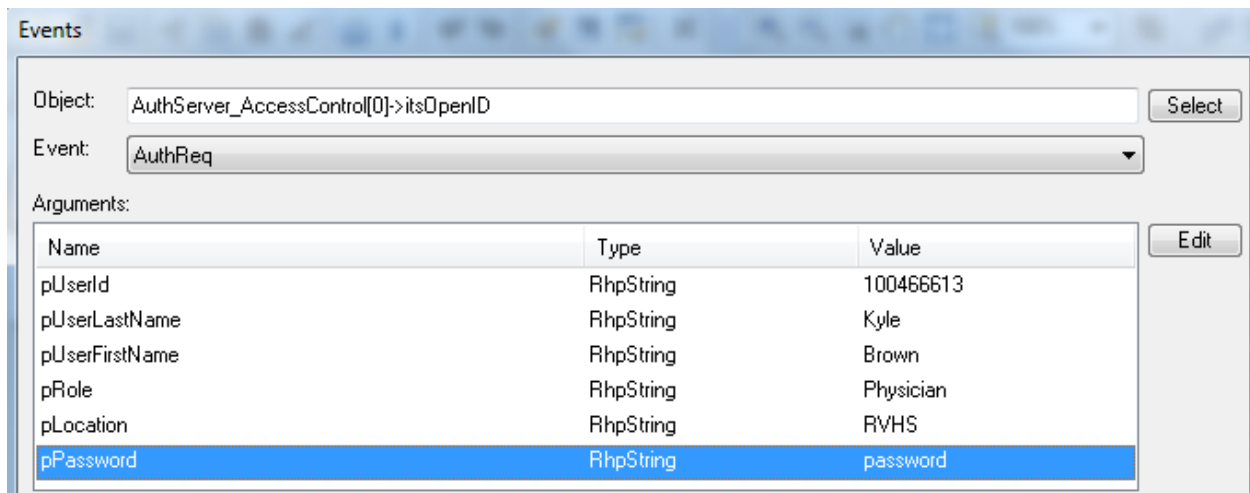
**Fig 3: Statechart of class OpenID**

As we see in Fig 4, initially OpenID module is in 'waiting' state. Then user sends an authentication request to OpenID module. User submits his credential and other details that have to be securely stored in OpenID database. This can include the user's personal details, work details, etc. In our scenario user enters his credential and work details, which includes his user id, password, name, role, location.

When OpenID receives the user id and password, it checks it with the data in the database to see if the user has already registered with it. If the user is already registered, it goes to 'user already exist' state and then it goes back to waiting state. On the other hand, if the user is new, OpenID module issues an openid id to the user.

In the actual implementation, this OpenID is a url issued to the user. In our scenario, we are generating an OpenID, which is a random id unique for each user. In the database user details are separated from user credentials and are stored in two different locations. This ensures that user data and credentials are more secure. We use the openid generated to link between two tables.

From OpenID Statechart in Fig 4, we see that user initially sends authentication request to the module. User is registered and the data is stored in the database. This will look like Fig 4, Fig 5 and Fig 6 given below:



The screenshot shows a software interface for viewing events. The 'Object' field contains 'AuthServer\_AccessControl[0]->itsOpenID' and the 'Event' dropdown is set to 'AuthReq'. Below, an 'Arguments' table lists the following data:

Name	Type	Value
pUserId	RhpString	100466613
pUserLastName	RhpString	Kyle
pUserFirstName	RhpString	Brown
pRole	RhpString	Physician
pLocation	RhpString	RVHS
pPassword	RhpString	password

**Fig 4: Authentication Request by user**

```

C:\Users\100466613\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch\User Pinned\Ta...
UserId : 100466613
Password : password
UserFirstName : Brown
UserLastName : Kyle
Role : Physician
Location : RVHS
Valid credential proceeding to register user....
User information saved!
Generating openId id....
OpenId Id generated and saved!!!
Congratulations!! You have been registered.

```

**Fig 5: System Response**

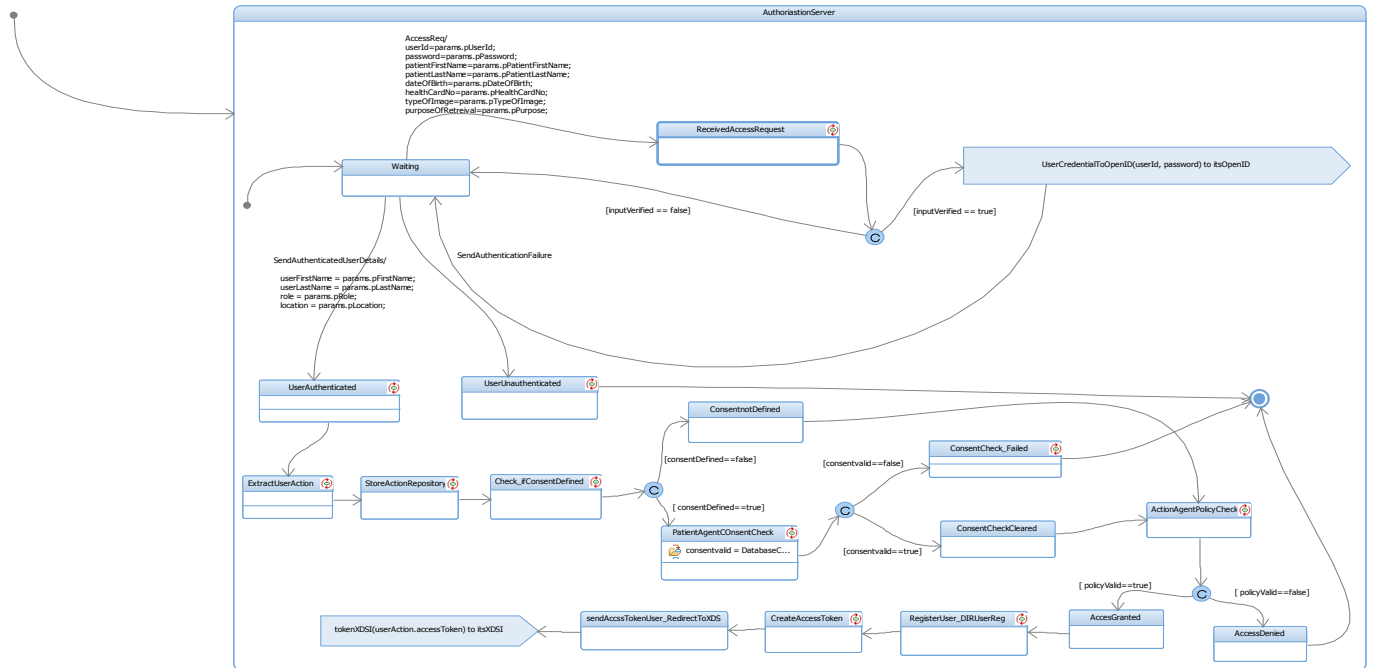
UserId	Password	FK_openID
10036613	pass	10036613openId
100466613	password	100466613openId
100466690	password	100466690openId

**Fig 6: Table with user credentials and OpenID in DB**

OpenID	UserFirstName	UserLastName	Role	Location
10036613openId	Mimi	Pooku	Physician	HHC
100466613openId	Brown	Kyle	Physician	RVHS
100466690openId	Anna	Krupa	Doctor	RVHS

**Fig 7: Table with user details and OpenID in DB**

## Step 2 : Access Control Check and Issuing access token



**Fig 8: Statechart Access Control Module**

'AuthServer\_AccessControl' class has the statechart as shown in figure 8. This module is designed to perform the following tasks:

- Receiving access request made by a user to retrieve the image of a particular user
- Checking to see if the user has entered the patient date of birth in the proper format
- Contacting the OpenID module and verifying the authentication of the user
- Retrieving user work details such as user role, location from the OpenID module if the user is found to be authenticated
- Storing the user action ( user details, access time, patient details, type of image and purpose of accessing) into action repository
- Checking if the access request made by the user is allowing the user to access the corresponding image based on the consent defined by the patient
- Checking if the access request made by the user is allowing the user to access the corresponding image based on the system security policies



- Assigning DIR user ID (DIR User ID is created by modifying the user ID) and registering the user with the DIR user registry once the user has successfully gone through all the access control checks.
- Issuing access token to the user. This token will have the DIR user ID encrypted in it.

## Snapshots from Simulation:

### 1. Access Request made by the user

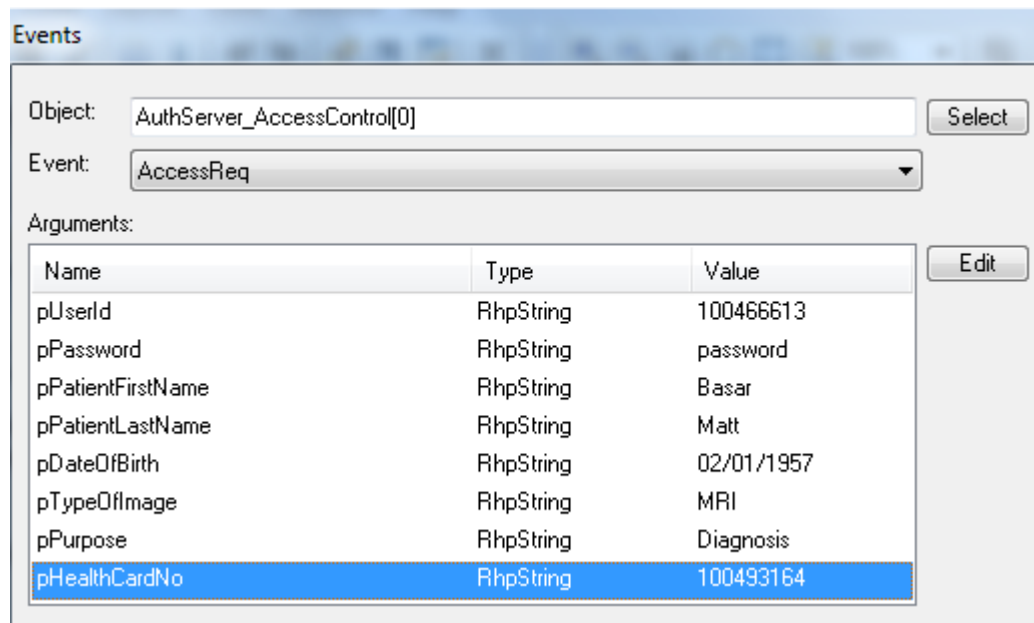


Fig 9: Access Request by user

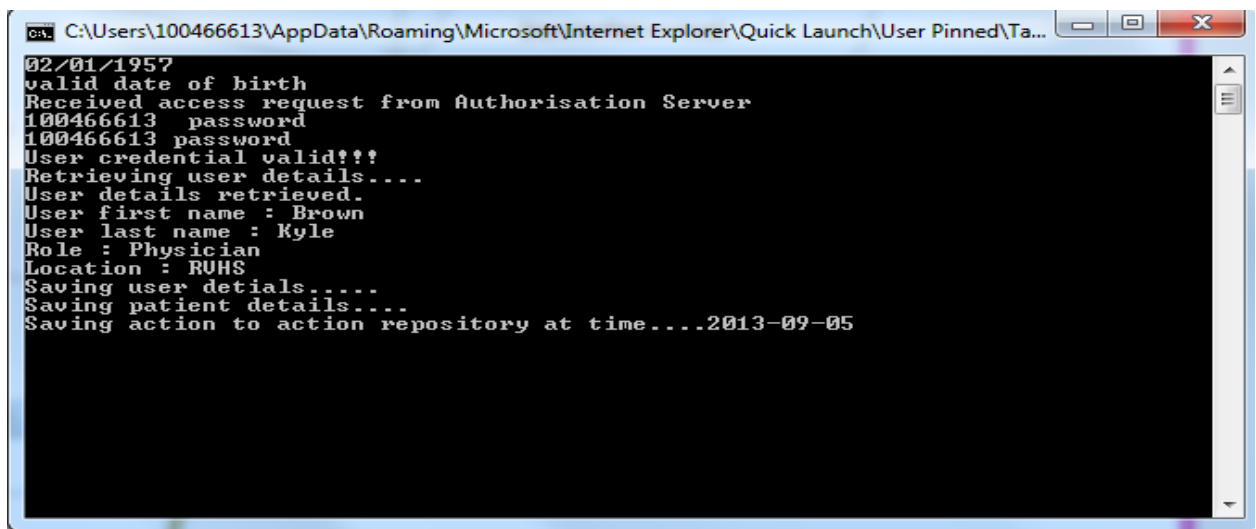


Fig 10: System Response to access request

## 2. User Action stored to action repository

As we see in Figure 11, access request made by the user is stored in the action repository. User details and patient details are stored in different tables and are linked to the action repository table using user ID and patient HCN respectively.

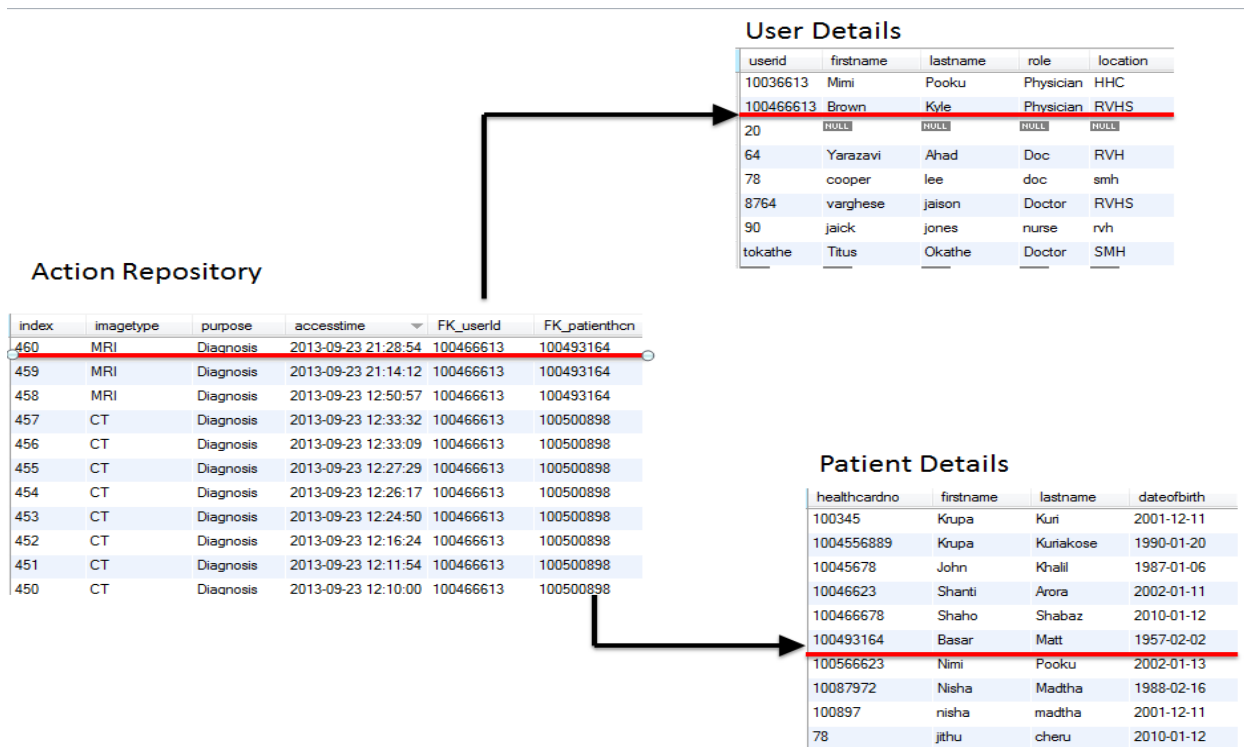


Fig 11: Snapshot of action repository storing access request made by user

## 3. Patient Agent : Comparing the user action with patient consent

Next, we check the patient consent repository to see if the patient, for this particular user, has defined consent. In figure 12, we see the patient consent repository. Consent is defined by a patient for different users holding different roles (doctor, nurse, lab technician, etc.). For example, let us consider data corresponding to index 1 in the 'patient consent repository'. We see (6, 1, 15, 15, 1) as values in row corresponding to index 1. This can be read as, patient whose details corresponds to index 6 in 'Patient Details' table has defined a consent for a user whose details corresponds to index 1 in 'User Details' table. Value 15 in column 'imagetype\_permission' corresponds to index 15 in 'Image

Type-Permission' table. Index 15 has values (1, 1, 1, 1) corresponding to column (MRI, CT, US, XRAY). These columns can take values 0 or 1. 1 corresponds to permission granted to access the image and 0 otherwise.

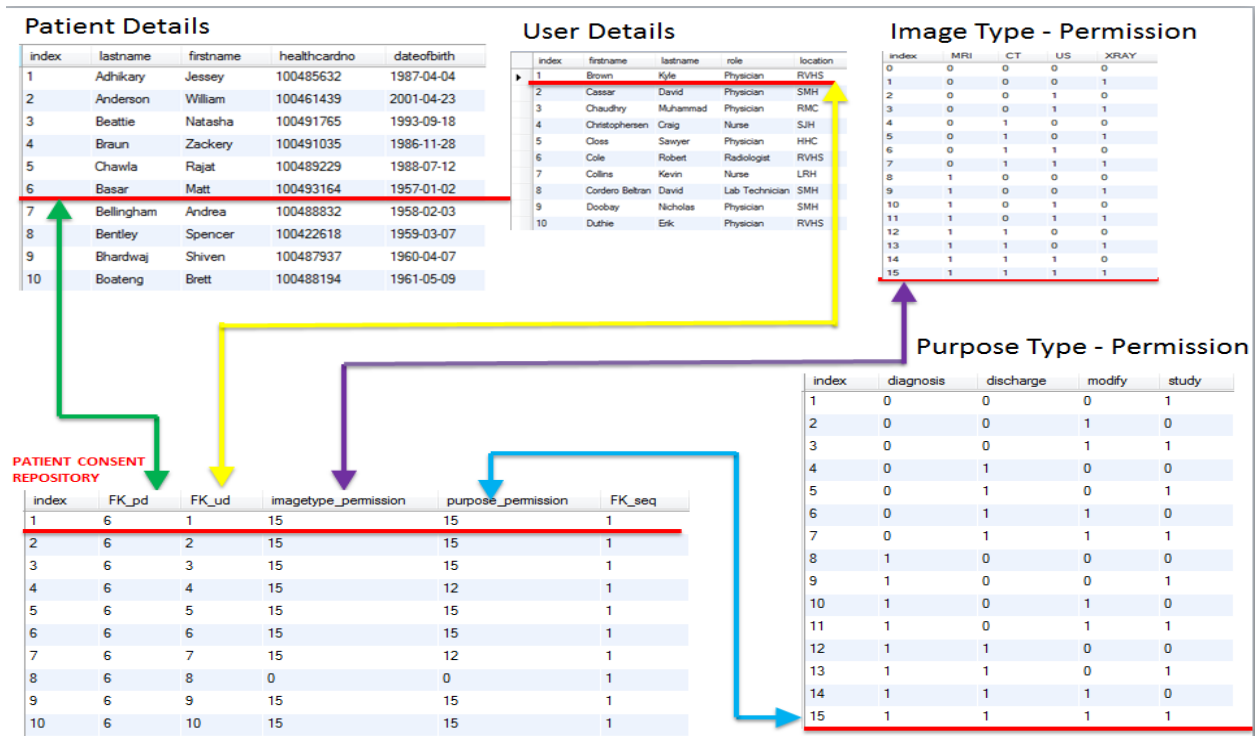


Fig 12: Snapshot of patient repository and associated tables

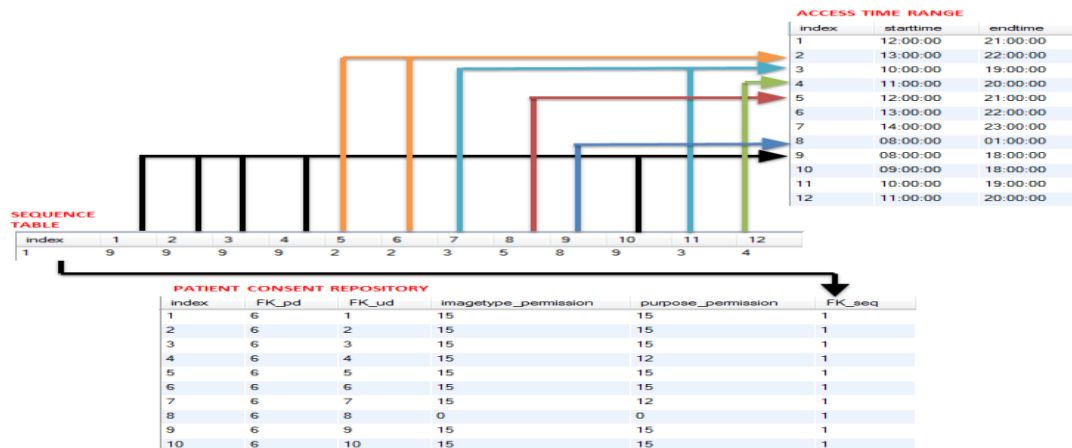
Similarly, Value 15 in column 'purpose\_permission' corresponds to index 15 in 'PurposeType-Permission' table. Index 15 has values (1, 1, 1, 1) corresponding to column (diagnosis, discharge, modify, study). These columns can also take values 0 or 1. 1 corresponds to permission granted to access image for that particular purpose and 0 otherwise.

```

Checking if the imagetype_permission and purpose permission is 1 or 0
ImagePermission : 15
PurposePermission : 15
segno : 1
MRI value : 1
CT value : 1
US value : 1
XRAY value : 1
1. Permission granted for the specified image type
diagnosis value : 1
discharge value : 1
modify value : 1
study value : 1
2. Permission granted for the specified purpose

```

Fig 13: Snapshot of system response based on the image type and purpose requested by the user



**Fig 14: Snapshot of patient repository and details about column 'FK\_seq'**

Last column 'FK\_seq' in 'patient consent repository' corresponds to the sequence selected by the user. We see that user has selected value for 'FK\_seq' column as 1. Value 1 corresponds to the sequence defined in 'sequence table'. We can define different sequences based in values from the 'access time range' table. Allowed access time range, which is the time interval during which user can access the data of a patient. We have defined the time sequence for each month. We can define sequence for weekly, bi weekly, yearly and so on depending on the choice of the system designer. In our scenario, we have decided to move on by defining access time range on a monthly basis as shown in Fig 13.

In our example, as you can see from Fig 11, action repository contains the record of time and date corresponding to the access request made by the user. We extract the month from the date and then extract the value in the 'sequence table' corresponding to the month. Then we check to see if the access time falls within the 'starttime' and 'endtime' specified in the 'access time range' table. Result of this check is showed in Figure 15.

```
Time and date value : 2013-09-23 21:28:54.0
month 09
month in integer 9
resultset :8
Seqno for the sequence defined by the patient for a particular user,accessing the
system at a particular date of a month :8
Retreiving the time interval corresponding to the sequence No:
Statement worked i guess!
access time raw data :9:28 PM
Retereving Start time and end time corresponding to the time access request was
made
Allowed range :- Start Time: 8:00 AM End Time : 1:00 AM Access Request Time:9:28
PM
Access Time is within range1
```

**Fig 15: Snapshot of system response after checking the access time**

By the end of this step user access request were compared with the consents defined by the patient to access his medical images (MRI, CT, US, XRAY) for certain purposes (diagnosis, discharge, modify, study) at certain time of the day. Depending on the permission granted to the user at the end of this stage, access request is forwarded to next stage for checking with the system security policies. If the user failed to meet the consents defined by the patient, he is no longer allowed to access the image and the request is terminated.

#### 4. Action Agent: Comparing user action with system security policies

The system security administrator defines system security policies for different people playing different roles (physician, nurse, lab technician, etc) in a hospital environment. In our example, Dr. Brown Kyle is a physician working in RVHS. We check with the 'system policy repository' to see what are the access privileges given for a physician from this location with respect to type of image and purpose of retrieve.

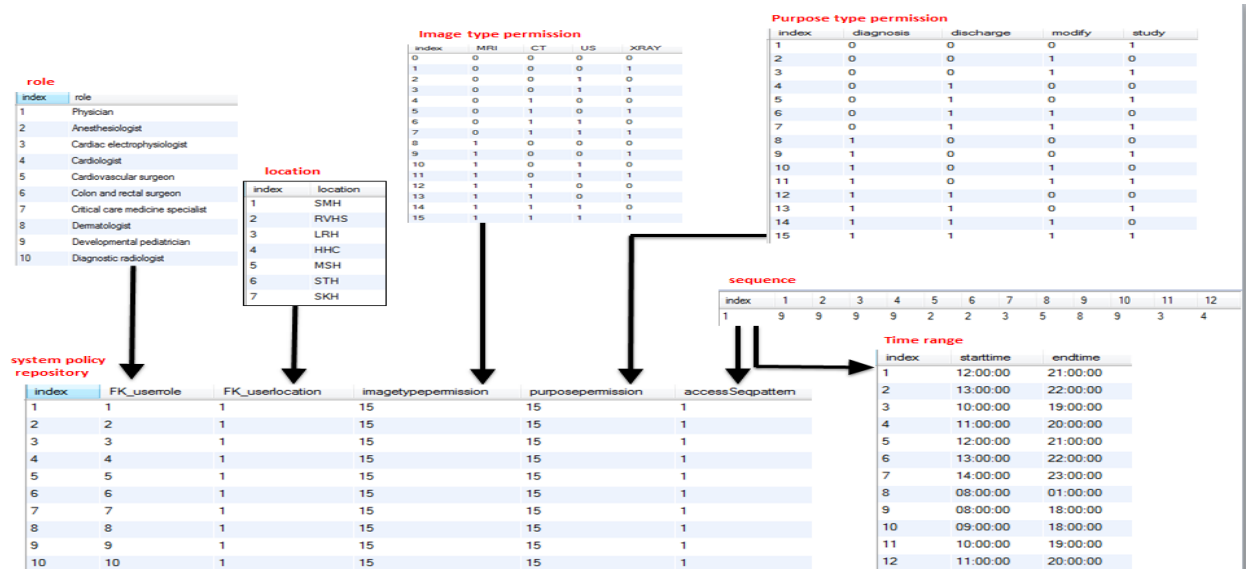


Fig 16 : Snapshot of system policy repository and associated tables

System policy check is very similar to patient consent check except for the fact that consent check is user specific whereas policy check is role specific. In figure 16, we see the system policy repository. A security administrator defines access policies for different roles (doctor, nurse, lab technician, etc.) that access the system from different medical environment. For example, let us consider data corresponding to index 1 in the 'system policy repository'. We see (1, 1, 15, 15,

1) as values in row corresponding to index 1. This can be read as "Physician from SMH can access MRI, CT, US and XRAY images for the purpose of Diagnosis, Discharge, modify and study". In our example, Dr. Brown Kyle is a physician working in RVHS has same privileges and hence his access request is processed and is successful. Figure 16. 'Accesssequencepattern' column works exactly the same way as described in the section above.

Once the user agent has success fully cleared the consent check and system policy check we issue DIR user ID to the user. This is discussed in the next section.

## 5. Assigning DIR user Id to user and storing it in DIR user registry

DIR user Id is dynamically assigned to the user once he has cleared the access control checks. In our example, Dr. Brown Kyle with user ID 100466613 has successfully passed all the checks and we assign DIR user ID in the following way:

$$\text{DIR User ID} = (\text{user ID} * 3) + 23 = 301399862$$

This is to generate an ID for the session. Each user will have a DIR user id and is stored in the DIR user registry by including the access time as well. Including time will help us to retrieve this ID at a later stage in an efficient way.

index	userid	diruserid	sessiontime
100466731	100466613	301399862	2013-09-23 23:03:16
100466630	100466613	301399862	2013-09-22 00:15:37
100466629	100466613	301399862	2013-09-22 00:11:08
100466628	100466613	301399862	2013-09-22 00:08:05
100466627	100466613	301399862	2013-09-22 00:06:58
100466626	100466613	301399862	2013-09-22 00:03:56
100466625	100466613	301399862	2013-09-22 00:01:45
100466624	100466613	301399862	2013-09-21 23:55:41

Fig 17: Snapshot of DIR user registry

## 6. Issuing access token to the user

Access token is usually a combination of numbers that are issued to the user as a key or permission to access the requested resources. In our model, we embed the DIR user ID within the token. This is to retrieve the user ID at a later stage. Token is generated in the following way and issued to Dr. Brown Kyle:

**token = 1234+ DIR user ID +5678 - (90\*78)\*23432 + (2356\*6785) + (55555\*767676) = 1437377470**

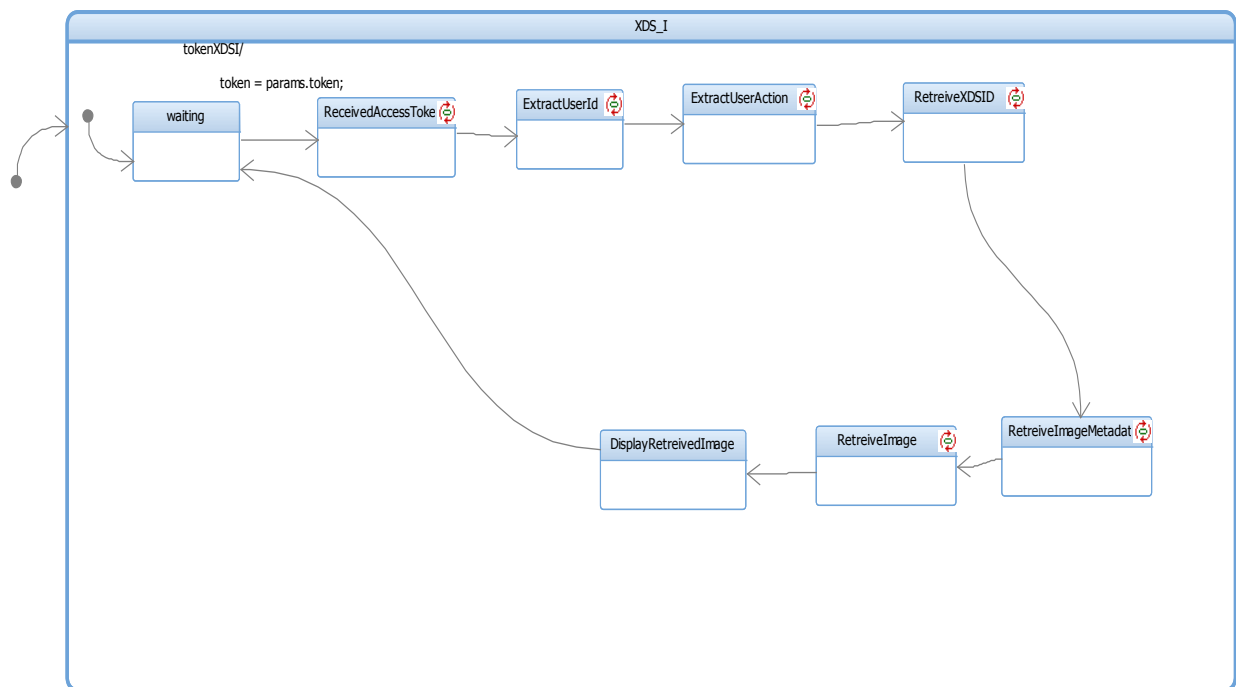
If the token issued to a user is lost, it is difficult for a third party to get any information related to the user or patient.

```
Creation OAuth Token
*****
userAction.userId : 100466613
userId in int : 100466613
DIR user id is : 301399862
Access Token is : 1437377470
```

**Fig 18: Snapshot of system response after creating access token**

At the end of this stage, the token is send to the user service module in the architecture shown in Fig 1. Later user approaches XDS-I module with this token to retrieve the image. In the following section, we describe the process of image retrieval.

### **Step 3: Retrieving Image using XDSI**



**Fig 19: State Chart XDS-I Module**

'XDSI' class has the statechart as shown in figure 19. This module is designed to perform the following tasks:

- Receiving the access token issued to the user
- Extracting DIR user ID by decrypting the token
- Extracting the original User ID from the DIR User ID
- Retrieving the user action (the access request made by the user) using user ID
- Finding the XDS ID of the patient using the Health card Number of patient specified in access request
- Using XDS ID to find image metadata and the DIR image number from XDS document repository
- Using Image number to retrieve image in DI-r

In our example, when the XDS-I module receives the access token, It is decoded to obtain the DIR user ID. We use this ID to extract the User ID.

$$\text{cipher} = (5678 - (90 * 78) * 23432 + (2356 * 6785) + (55555 * 767676) + 1234)$$

$$\text{DIR user ID} = \text{token} - \text{cipher}$$

Further we use the User ID to retrieve the access request made by the user (user action) stored in the action repository. User action contains the details of patient (name, health card no, date of birth), type of image requested and the purpose of retrieving the image. Health Card Number (HCN) is the unique identity for each patient. Hence, we issue XDS ID for patients and map them to the HCN. Later we use the XDS ID to find the image Metadata stored in 'XDS document repository'. From 'XDS Document repository', we get the metadata associated with the image. This includes the patient details (name, HCN, DOB), author details (name, role, location), exam details (modality, body part, procedure code, location, date) and image no (corresponding to image in DIR).

In our example, Dr. Brown Kyle wants to access the MRI image of patient Matt Basar with Health Card No: 100493164, DOB: 1957-01-02 for the purpose of diagnosis. This data is received from the action repository. From Fig 19, 'XDS Document Registry' we see that XDS ID corresponding to HCN 100493164 is 811393833. Index 6 of 'Patient Details' table corresponds to the details of patient. Index 6 'Exam Details' table gives the details of MRI image. Index 6 of 'Author identity' table gives the details of person who took the image.



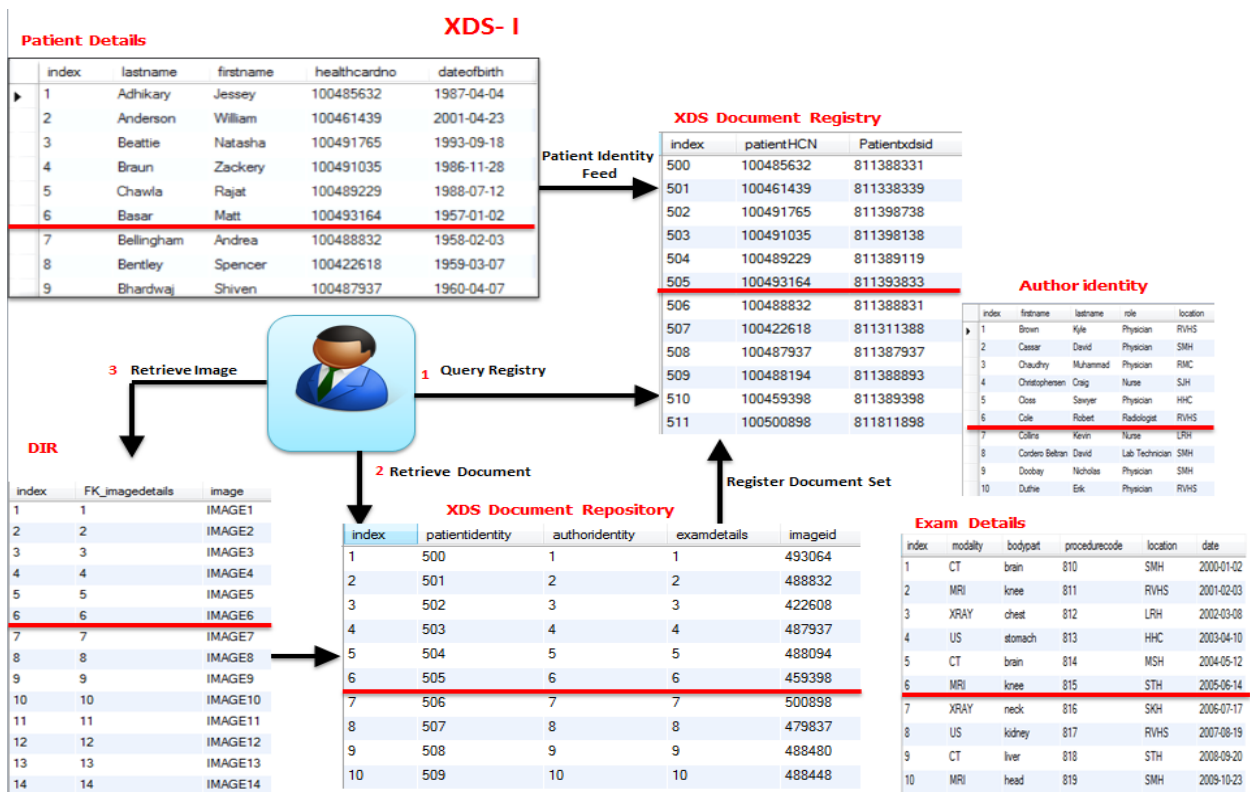


Fig 20: Snapshot of various tables involved in XDSI

Once we get the image metadata, we retrieve the image from the DI-r.

```

Extracting DIR user ID from the token :
diruserid decrypted from token: 301399862
UserID corresponding to the diruserid decrypted from token : 100466613
State: ExtractUserAction
HCN of Patient specified by user in access request 100493164
State: RetrieveXDSID
***** Retrieving patient XDSID from XDS Document Registry*****
Patient XDSID : 811393833
State: RetrieveImageMetadata
***** Image Metadata from XDS Document Repository *****
Modality      : MRI
Body Part     : knee
procedurecode : 815
Location      : STH
Exam Date     : 2005-06-14
Dir ImageNo   : 459398
State: RetrieveImage
*****Displaying the image stored in DI-r*****
Image: IMAGE6

```

Fig 21: Snapshot of system response to XDS-I module

