# FionaDocs

**Hauke Bartsch, Marek Kociński**

**Jul 28, 2025**

# LEARN

FIONA (Flash-based Input/Output Network Appliance) - A secure research data gateway for medical imaging. Provides DICOM anonymization, quarantine management, and automated transfer from clinical to research PACS systems while ensuring GDPR compliance.

# ENDUSER

**For:** Doctors, researchers, medical personnel

## 1.1 How to apply to the Research Information System Helse Vest

### 1.1.1 1. Project creation, setup and access

In order to apply for a new project on the research information system (research PACS) please fill out the application form available under "Apply/Apply for a new research project" here: https://www.google.com or ttps://fiona.medtek. hbe.med.nvsl.no/.

Additional user access can be requested by the principal investigator of the project under "Apply/Apply for access to an existing project".

If you encounter any problems with applying for access, contact Hauke.Bartsch@helse-bergen.no.

### 1.1.2 2. Access to REDCap for structured data

Our project uses REDCap as an electronic data capture solution. Projects on the research information system can receive access to their REDCap project as well as access to the image data viewing (see next section).

### 1.1.3 3. Access to the "Sectra DMA Forskning" research PACS viewer

Access to the image data is provided by IKT. Such access requires a valid Haukeland University Hospital user account and a laptop or PACS workstation that is under control of IKT. If you contact IKT ask for the start menu item "Sectra DMA Forskning". With the program and your hospital username and password you will gain access to the research picture archive and communication system (PACS).

Without access to a specific research project you will not see any data in the research PACS. Each research projects requires specific permissions to become accessible for a user.

The research PACS viewer is using a separate clinical PACS software installation (Sectra IDS7). In order to prevent possible interactions between the clinical and the research PACS only one of the application can run at a given time. You will be logged out of the clinical PACS if you start the research PACS viewer.

### 1.1.4 4. Submit data to the Research Information System

Overview: The research information system contains two components. Image data is stored in the Sectra DMA Forskning - an image viewer with a vendor neutral archive (VNA). All meta-data is stored in table format in an electronic data capture system (REDCap). Sending image data will create the appropriate entries in REDCap. Additional data collection instruments can be set up there and used to capture assessments, consent/assent and results from automated image processing. All image data is assigned to a project to allow for project specific data views for each research information user.
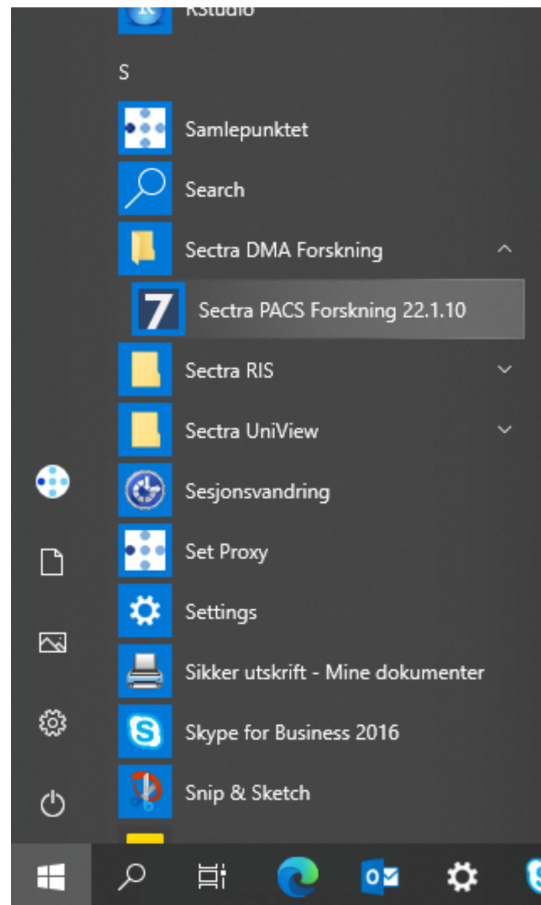
The basic steps to submit data are:

Fig. 1: Forskning PACS start menu icon required to view image data by project.

1. Send DICOM studies to "HBE Fiona" or "Fiona" (modality station)

2. Assign to project on https://fiona.medtek.hbe.med.nvsl.no/applications/Assign/

In step 1 data arrives in a **quarantine** location. In step 2 each DICOM study needs to be **assigned to project**, pseudonymized participant identifier and event name before it will be forwarded to the research PACS and becomes visible to the project users.

## 4.1 Setup of a new project

The project needs to exist on the research information system before participant data is collected. After a successful setup your project and event names should appear in the Assign application.

## 4.2 How to add image data

The end-point for images is FIONA (https://fiona.ihelse.no):

- AETitle: FIONA

- IP: 10.94.209.30

- Port: 11112

Images that arrive at this endpoint are added to a quarantine system (FIONA, https://fiona.medtek.hbe.med.nvsl.no:4444) running the REDCap software. Automatic routing rules (stored in REDCap) are used to anonymize and forward the data to the image storage. If such routing has not been set up the "Assign" application (see below) needs to be used to forward individual studies based on pre-existing patient ID lists.

From Sectra Production you can send image data to the endpoint "HBE Fiona". Modality stations might also have the "FIONA" endpoint setup. If the data is already anonymized and has a de-identified PatientName/PatientID entry that indicates the project the FIONA system will attempt to de-identify (pseudonymization) further DICOM tags and forward the images to IDS7 (may take minutes). No further action is needed. If you suspect this did not work, see the corresponding section about the representation of transfers in REDCap.

Image data that contains patient information cannot be automatically assigned to the appropriate project as there is only a single endpoint for FIONA shared by all projects. To assign participants correctly to projects and de-identified participant identifiers a user can perform the assignment to project, participant ID and event name in the "Assign" web application.

If the participant identifiers do not exist yet user may add new project specific identifiers in "Assign". Such identifiers need to follow the naming rules for a project and are verified using regular expression pattern specific for each project.

The web application for the assignment of scans forwarded to HBE Fiona is available at: https://fiona.medtek.hbe.med.nvsl.no/applications/Assign/

On the Assign website look for your forwarded study. It should appear in about 15 min. Identify the correct scan using the Accession Number (Undersøkelse-ID) or the date and time of the scan. Select your project from the drop-down. This will fill in the list of patient names and event names. Select the correct patient name and the event this study belongs to. After a couple of seconds a new button appears below the study entry. Use it to select and confirm the assignment. This will forward a de-identified version of the study data to "Sectra Forskning". If you do not assign your data on Assign they will not be forwarded. After a couple of days (7 days) such data will disappear from the list. Send an email to Hauke to request a resend.

### 4.2.1 Verification steps

After data arrived at the research PACS a verification step should ensure that all images have been received at the quarantine on FIONA and have been forwarded to research PACS. This can be done by comparing the number of images on the sending station with the number of images in IDS7.

Furthermore the import step will also attempt to de-identify secondary capture images with burned in image information. This process is fully automated and can result in false positive and occasionally false negative results. After a

review of the data in IDS7 the user may decide which secondary image series are "safe" to exclude from the pixel rewriting on import. For example a secondary capture series from DTI may not contain any burned in names or identifying numbers or dates. Such image series can be removed in REDCap from further pixel anonymization.

If the number of images on FIONA does not correspond to the number of images available cache previous assignments and automatically forward such images to the research PACS using the reviously defined project, patient identifier and event name.

## 1.1.5  5. Export image data from research PACS

Data in the research PACS is secured by generic procedures during data import that delete or rewrite some DICOM tags, changes dates and replaces unique identifiers. A documentation of this process is available on the GitHub repository of the projects for removal of DICOM meta-tags: https://github.com/mmiv-center/DICOMAnonymizer, and for the removal of burned in image information: https://github.com/mmiv-center/RewritePixel.

Data stored in the research PACS is therefore in general suited for data sharing IF pseudonymized data is allowed. In order to support users with the task of data pseudonymization the research information system provides the "Review" web application that lists all existing DICOM tags in a research project (https://fiona.ihelse.net).

> **Note**
>
> Pseudonymized data is defined here as data for which a coupling list exists somewhere in the universe. This is in contrast to anonymized data where such a list does not exist and can also not be created.

Further de-identification procedures might require changes to image data such as face stripping, removal of outer ear tissue, cortical folding pattern, etc.. Such potential sources of information for re-identification have been proposed in the literature but actual attacks based on them have not recently been documented. Better documented and perhaps more relevant are re-identification using spreadsheet data where external sources are linked to the projects data to discover the supposedly hidden identity of the research participants. For example it might be possible to link Gender, day of birth and the hospital name to a real participant name using a birth or voting registry.

### 5.1 Export using IDS7

The image data from a study can be exported from the research PACS using a right-click menu entry available in the Informasjonsvindu "Exporter til medium". Such exports will generate either a derived patient ID – if an Anonymization Profile is selected or a faithful copy of the data with all pseudonymized DICOM tags intact.

> **Note**
>
> This export does not prevent re-identification. Specifically the PatientID field is created from the pseudonymized ID used in the research PACS and therefore not random.

The export is also case-by-case, which is tedious if many data need to be exported. The export will also result in directory names that do not reflect the research project structure as participant identifier – event name – modality – image series. It may be advantageous to export from IDS7 if a single image study needs to be shared without special requirements. Such export folders will also contain an image viewer.

### 5.2 Export to project specific formats, NIfTI and zip-files

The research information system supports a separate export facility that is more suited to implement project specific de-identification. Such export requirements include specific DICOM value changes (replacing underscores with dashes), adding birth date information back, formatting and cleaning of series descriptions, zip-file exports with specific folder structures etc.. This export is appropriate if the receiving institution has specific requirements on how data should be shared.

Request access to the specialized data exports for your project from Hauke.Bartsch@helse-bergen.no. Provide your export specification and we will implement your anonymization scheme and make it available to you and other researchers. As an example the "Export" application currently supports the export in NIfTI formats (using dcm2niix) and the export in several zip-file formats.

### 1.1.6  6. Research Information System

The research information system (RIS) of the Western Norway Health Authorities (Helse-Vest) also called the "Steve Project" is a secure computer system that stores research data for approved research projects at Haukeland University Hospital and connected hospitals of the Helse Vest region. The project is supported by the radiology department of Haukeland University Hospital and the Mohn Medical Imaging and Visualization Center and approved for research project use by IKT Helse Vest. The physical location of the data is at the premises of IKT Helse Vest Norway. Dedicated storage area and research software (Sectra, IDS7) provides researchers with appropriate permission access to their data. All data is stored in a de-identified format inside the RIS. Maintaining a coupling list is the responsibility of each project and not part of the functionality of the RIS.

Based on the REK/DIPA rules for each project a lifetime tracking of the research data per project ensures that data can be anonymized based on data sharing requirements, and that data can be deleted at the end of the project phase - if required. We suggest that research data is allowed to be fully anonymized at the end of the project and remain in RIS for general research access.

Key features of the RIS include:

- Hosted side-by-side with the clinical PACS as an independent installation.

- Accepts de-identified patient identifiers only.

- All data is moved through a de-identification process upon import into RIS.

- All data is assigned to one or more specific research projects and the visibility of data is restricted to individuals with project role access rights.

- Projects require a valid REK approval, such documentation has to be provided at the start of a project by the project owner.

- The project owner can identify additional user accounts that can access the data.

- User access to the research PACS is controlled by IKT and requires a valid Haukeland University Hospital user account.

# ADMIN

**For:** IT administrators, DevOps

## 2.1 End-user contract

> **Note**
>
> The following text is from the Apply website for the Steve system. Please check this page for updates to the wording.

By creating a project on the research information system you agree to the following:

1. All data stored in the RIS belongs to the research project owner represented by the PI of the project. Adding and verifying added data to the RIS is the responsibility of the project owner. The RIS team will help research projects to automate this process.

2. Sensitive participant information needs to be stored under a separate account and needs to be accessible to authorized (data-manager and above) user accounts only. All other research data is stored as de-identified data (pseudonymized, with external coupling list) or in an anonymized format. This restriction includes sensitive information such as Norwegian identification numbers, real names or parts of real names, other birth certificate information and initials. It is the responsibility of the project to review the result of the de-identification procedures implemented by the RIS team on image meta-data using https://fiona.medtek.hbe.med.nvsl.no/applications/ReviewDICOMTags and the result of the detection and removal of burned in image data (IDS7). The project will inform the RIS team in a timely manner if the pseudonymization procedure of the RIS team needs to be updated. This restriction is in place to allow for the largest possible user base for the RIS including PhD students and external collaborators.

3. All research data is stored as part of RIS projects identified by a project name of 5-20 characters. Users can gain access to the data upon request from the project PI or an appointed representative of the PI.

4. Projects are expected to utilize best-practises for data handling such as accounts based on roles like data-entry (add data only) and data-manager (change data, export data). Personally identifying fields have to be marked as such (Identifier? field of the instrument designer) and data access groups shall be used for multi-site project setups.

5. Projects will undergo a short review from the RIS team before they are moved by the RIS team from development mode into production mode for data capture. This review may generate suggestions for the project on how to implement best practices for longitudinal data captures, missing validation and the use of additional software features. All research data is collected and stored with a valid REK approval for the time period specified in the REK approval. The REK approval is required at the time that the RIS project is created. Any change of the REK approval start and end dates need to be reported to the RIS team. At the end of the project period data can be either: a) deleted or b) fully anonymized (suggested choice). It is up to the project to inform the RIS team about the correct way of handling the data at the end of the project. By default we will assume that data needs to be deleted. Based on the project end date (REK) the RIS team will inform the PI of the project of a pending change

of the project status to the archive state. An archive state project will not allow for further data entry, or changes to captured data. After a period of about 1 year the project data will be exported and provided to the project PI for download. An archived project can be deleted by the RIS team after an unspecified time period. If the project data can be fully anonymized, the RIS team may create a copy of the data with new participant identifiers (without a coupling list). After a re-import a fully anonymized version of the project data can become accessible to other RIS users. The original project data will change to archive state, a copy is provided to the projects PI and the data can be deleted by the RIS team after about 1 year.

## 2.2 Features for data migration

The Assign web-application allows users to upload a coupling list that maps the accession number (Undersøkelse-ID) of the study to the pseudonymized participant identifier. Such mappings must be uploaded before the first image study of the project has been forwarded to FIONA. Incoming DICOM studies in FIONA that match entries in the coupling list will automatically be assigned to the project.

## 2.3 How to handle errors?

Correcting errors during data import are not difficult to fix. Try to follow up on such errors on an ongoing basis. The quarantine FIONA station may have still have a copy of the data in its cache which simplifies the process. Contact Hauke.Bartsch@helse-bergen.no in such cases and ask for help. This will allow you to fix issues such as:

- Wrong assignment of participant identifiers to DICOM studies

- Wrong assignment of event names to DICOM studies

- Missing images or image series for existing DICOM studies

- Missing entries for DICOM studies on "Assign"

## 2.4 Export to project specific formats, NIfTI and zip-files

The research information system supports a separate export facility that is more suited to implement project specific de-identification. Such export requirements include specific DICOM value changes (replacing underscores with dashes), adding birth date information back, formatting and cleaning of series descriptions, zip-file exports with specific folder structures etc.. This export is appropriate if the receiving institution has specific requirements on how data should be shared.

Request access to the specialized data exports for your project from Hauke.Bartsch@helse-bergen.no. Provide your export specification and we will implement your anonymization scheme and make it available to you and other researchers. As an example the "Export" application currently supports the export in NIfTI formats (using dcm2niix) and the export in several zip-file formats.

## 2.5 Sensitive Data Projects – Separation of Sensitive Information and Data

A sensitive data project is one that is used to capture human subject data and in general will require a REK (regional ethics board approval). In order to setup such a project in REDCap we suggest the follow structure and features of REDCap to be used. These recommendations have been generated based on discussions in relevant risk assessments.

All sensitive data should be stored in a separate REDCap "ID" project including Norwegian Identification Numbers, names or parts of names, addresses and full birth dates (see Figure 1). This project should have its own roles of "Data Manager", "Data Entry", and "Controller". eople with permission to access and/or edit this information can use this database to keep contact information up-to-date and to enroll new participants into the study. Each participant should be assigned a pseudonymized ID in the sensitive data project that links the entry to the corresponding participant in the

data project. Examples for this ID are: <project name>-<site number>-0001, <project name>-<site number>-0002, etc..

All other data should be stored in a separate REDCap "Data" project using the pseudonymized participant ID as a "record_id" (first field in the study).
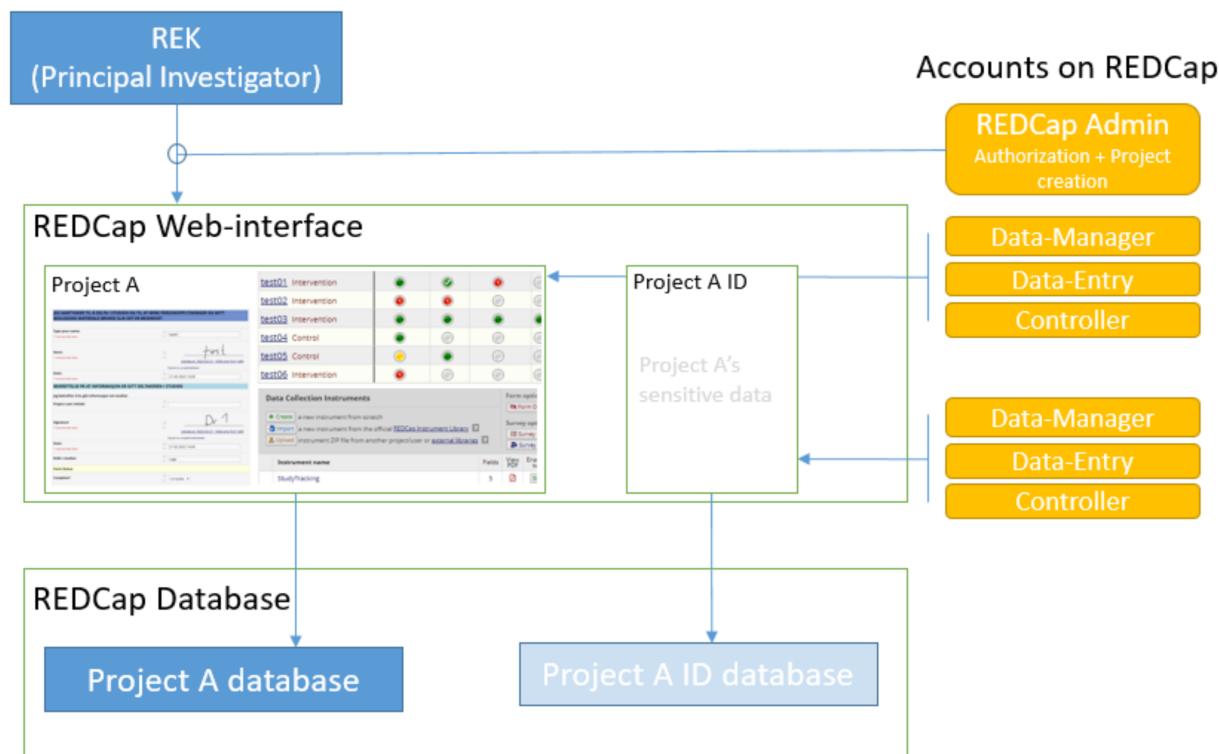


Fig. 1: Sensitive data projects should be split into a REDCap project for data (using pseudonymized ids) and a REDCap project for sensitive data including the coupling list.

### 2.5.1 User rights management

When a project leader / principal investigator (PI) is given a REDCap account and project, they are given "project owner" roles. The project owner can then provide access to project members in "roles". A role defines a given set of custom permissions which defines the user's access to data, export permissions and ability to make changes to data.

Each project can have predefined roles. We recommend the predefined roles "Data Manager" (ability to change study design, export), "Data Entry" (add, change, or delete data) and "Controller" to define roles for data viewing, editing, and deleting records. In more complex cases, different access settings can be given on different forms in the study (see also API access with REDCap). Individual users are assigned to project roles as part of gaining access to one project.

The user rights management is the responsibility of the project owner and/or the users they add to the project with User Rights access. User roles should be set at the lowest access level that is necessary (e.g., export rights only for users who need this permission). Access to the project should be reviewed regularly and personnel who no longer require access need to be removed from the project.

### 2.5.2 User rights – multi-center projects

In a project where several institutions participate with their own project participants (several hospitals etc.) each group of participants should be assigned to a separate "data access group". This functionality allows records in a study to be part of the user rights management. A user with access to a single data access group can only see participants that belong to this group. If this user creates a new participant, they will be automatically assigned to the group.

---

### 2.5.3 How to handle Email Addresses in Data Projects

Email addresses are special identifying fields that can be stored in data projects for the purpose of creating automated invites for participants to fill out forms from home. In projects that use this feature email fields need to be present in the data project in order to allow for email distribution to participants.

1. Add such email fields to a separate instrument of the REDCap data project and mark the instrument as viewable by specific roles only (like Data Managers).

2. Mark the email field as an "Identifier" field to prevent export of the field's data by user of roles that cannot view sensitive fields.

3. Add the Action Tag "@PASSWORDMASK" to the field to prevent accidental viewing of the fields values if the instrument is displayed on screen.

4. Add a field validation as "Email" to prevent some miss-typing of emails.

### 2.5.4 Automatic data exports from REDCap

Data may be exported from REDCap using the REDCap API, a technical interface to automate the export of project and participant information using scripting. To provide such access a dedicated user-account "api_<real username>" should be created which is specific for a single project. Configure the account with a limited set of read permissions to specific fields or instruments using a new API role. The REDCap API will borrow these restrictive permissions for controlled access.

Setup: An administrator can generate an API "token" for this account and share the token and examples of accessing the resource (curl-based access) with the user.

Any change in the role of the <real username> should also apply to the connected API account. Specifically loosing access to the project should be implemented for both <real username> and api_<real username>.

### 2.5.5 Steps at the end of a REDCap project

REDCap is a tool for data collection. At the end of data capture projects using REDCap receive a notification of study end. At this point projects may provide updated REK information(extension of data capture notice). If no such notice is received REDCap projects will:

- Lock all data participants (no further update/add).

- Provide a copy of the REDCap project (CDISC format) to the project's principal investigator or delegate.

- Provide a copy of the project data (CSV) and data dictionary (PDF) to the principal investigator or delegate.

- Request a confirmation that project data (CDISC and CSV) have been received by the project.

- Permanently delete all project data.

This process will be documented in the REDCap project tracking project "DataTransferProjects", the project management tool with information on identity of the person requesting project removal and confirmations for all steps of the project removal process.

Fig. 2: End-of-project tracking for REDCap projects

# ARCHITECTURE, DATA FLOW AND SYSTEM COMPONENTS

**For:** Developers, system architects

## 3.1 Architecture Overview

The architecture of Fiona sysetem can be included included into a few layers: network layer, processing layer, storage layer, transfer layer and management layer
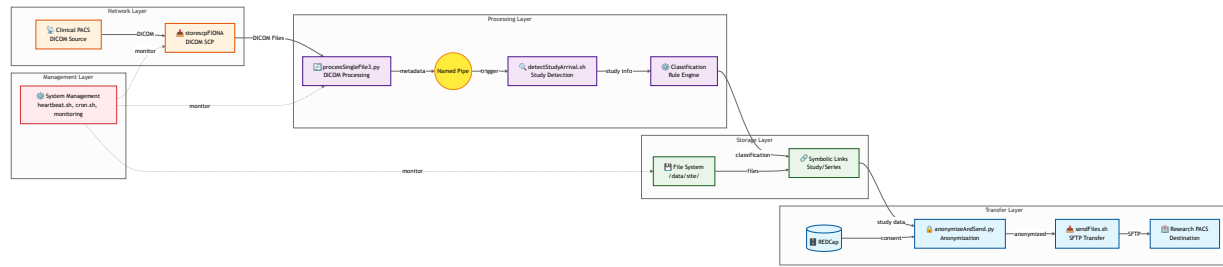
### 3.1.1 System Purpose

FIONA serves as an intermediary system that:

- Receives medical image data from clinical PACS systems

- Processes and classifies incoming DICOM studies

- Anonymizes data according to research requirements

- Manages data transfer back to research PACS systems

- Provides project-specific data organization

## 3.2 Data Flow

This document describes the complete data flow through the FIONA system, from initial DICOM reception to final transfer to research PACS.

Data flow overwie (ver.1 - detailed)

Data flow diagram (ver.2 - more general)

## 3.2.1 Data Flow Overview

```
┌─────────────────────┐
│   Clinical Systems  │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   DICOM Reception   │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   File Processing   │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Classification   │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│  Study Organization │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Anonymization    │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Transfer Prep    │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Research PACS     │
└─────────────────────┘
```
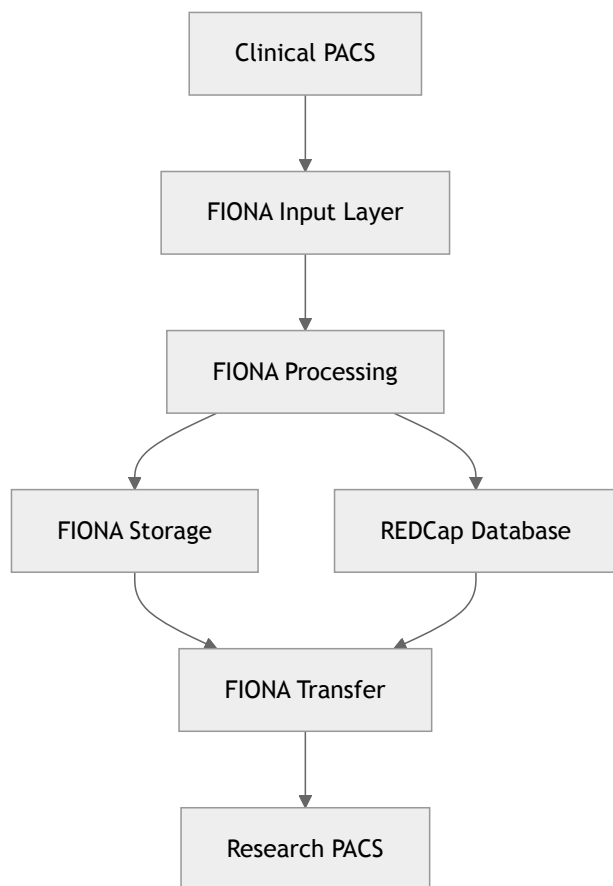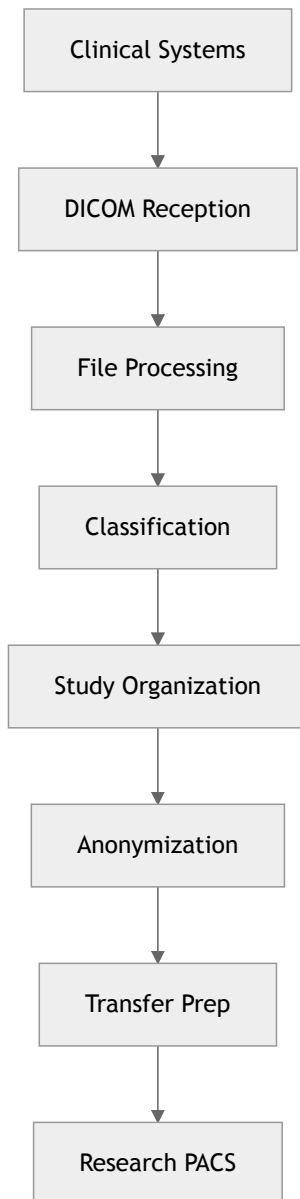
The FIONA system processes medical image data through several distinct phases:

1. **Data Reception** - DICOM files arrive from clinical systems

2. **Initial Processing** - Files are processed and classified

3. **Study Organization** - Data is organized into study/series structure

4. **Anonymization** - Data is anonymized for research use

5. **Transfer Preparation** - Data is prepared for transfer

6. **Export** - Data is transferred to research PACS

### 3.2.2 Detailed Data Flow

#### Phase 1: Data Reception

**Input:** DICOM files from clinical PACS **Components:** storescpFIONA, storectl.sh **Output:** Raw DICOM files in temporary storage

```
Clinical PACS → storescpFIONA → /data/site/.arrived/
                                ↓
                     Named Pipe (/tmp/.processSingleFilePipe)
```

**Process:** 1. Clinical PACS sends DICOM files via DICOM protocol 2. storescpFIONA receives files and stores in */data/site/.arrived/* 3. File arrival notification sent via named pipe 4. Files moved to */data/site/archive/* for processing

#### Phase 2: Initial Processing

**Input:** Raw DICOM files **Components:** processSingleFile3.py, receiveSingleFile.sh **Output:** Processed DICOM files with metadata

```
/data/site/archive/ → processSingleFile3.py → /data/site/raw/
                      ↓
            Classification Rules (classifyRules.json)
                      ↓
            Study/Series Organization
```

**Process:** 1. processSingleFile3.py daemon receives file notifications 2. DICOM headers are parsed and metadata extracted 3. Files are classified using rule-based system 4. Study and series information is organized 5. Symbolic links are created for easy access

#### Phase 3: Study Organization

**Input:** Processed DICOM files **Components:** detectStudyArrival.sh **Output:** Organized study structure

```
/data/site/raw/ → detectStudyArrival.sh → Study Job Directory
                  ↓
          Study Completion Detection
                  ↓
          Workflow Trigger
```

**Process:** 1. detectStudyArrival.sh monitors for completed studies 2. Study completion is detected when all series arrive 3. Study job directory is created 4. Workflow processes are triggered

### Phase 4: Anonymization

**Input:** Organized study data **Components:** anonymizeAndSend.py, anonymize.sh **Output:** Anonymized DICOM files

```
Study Data → anonymizeAndSend.py → Anonymized Data
                     ↓
        REDCap Configuration
                     ↓
        Project-specific Rules
```

**Process:** 1. Transfer requests are read from REDCap 2. Project-specific anonymization rules are applied 3. DICOM tags are modified according to requirements 4. Anonymized files are prepared for transfer

### Phase 5: Transfer Preparation

**Input:** Anonymized study data **Components:** createTransferRequest.py, createZipFileCmd.php **Output:** Transfer-ready data packages

```
Anonymized Data → createTransferRequest.py → Transfer Package
                        ↓
             ZIP File Creation
                        ↓
             MD5 Checksum Generation
```

**Process:** 1. Transfer requests are processed 2. Data is packaged into ZIP files 3. MD5 checksums are generated for integrity 4. Transfer packages are prepared

### Phase 6: Export

**Input:** Transfer packages **Components:** sendFiles.sh **Output:** Data transferred to research PACS

```
Transfer Package → sendFiles.sh → Research PACS
                        ↓
             SFTP Transfer
                        ↓
             Transfer Confirmation
```

**Process:** 1. SFTP connection established to research PACS 2. Files are transferred with integrity checking 3. Transfer status is logged 4. Success/failure notifications are sent

### 3.2.3 Data Storage Structure

The FIONA system uses a hierarchical storage structure:

```
/data/
├── site/
│    ├── .arrived/         # Initial file reception
│    ├── archive/          # Raw DICOM storage
│    ├── raw/              # Processed DICOM files
│    └── output/           # Processing results
├── config/                # Configuration files
└── logs/                  # System logs
```

Project-specific directories follow the pattern: /data{PROJECT}/site/...

### 3.2.4 Communication Mechanisms

**Named Pipes:** - */tmp/.processSingleFilePipe* - File processing notifications - Project-specific pipes: */tmp/.processSingleFilePipe{PROJECT}*

**Configuration Files:** - */data/config/config.json* - Main system configuration - *classifyRules.json* - Classification rules - REDCap integration for transfer management

**Log Files:** - System logs in */var/www/html/server/logs/* - Processing logs in */data/logs/*

### 3.2.5 Error Handling and Recovery

**File Processing Errors:** - Failed files are logged and can be reprocessed - Corrupted DICOM files are quarantined - Processing retries are implemented

**Transfer Errors:** - Failed transfers are retried automatically - MD5 checksum verification ensures data integrity - Transfer status is tracked in REDCap

**System Recovery:** - Daemon processes can be restarted automatically - File system consistency is maintained - Backup and recovery procedures are in place

### 3.2.6 Monitoring and Logging

**System Monitoring:** - heartbeat.sh - System health monitoring - cron.sh - Scheduled task management - Log rotation and management

**Data Flow Monitoring:** - File arrival detection - Processing status tracking - Transfer completion monitoring

This data flow ensures reliable, automated processing of medical image data while maintaining data integrity and compliance with research requirements.

## 3.3 System components

### 3.3.1 storectl.sh

The `storectl.sh` script is a system service controller for managing DICOM storage operations. It starts/stops the storescpFIONA daemon which receives DICOM files over the network and processes them through a named pipe system.

**The Main Dependecnes:**

- **user:** processing
- **depends-on:**
    - /data/config/config.json,
    - storescpFIONA,
    - receiveSingleFile.sh,
    - processSingleFile.py
- **log-file:**
    - ${SERVERDIR}/logs/storescpd${projname}.log,
    - ${SERVERDIR}/logs/storescpd-start.log
- **pid-file:** ${SERVERDIR}/.pids/storescpd${projname}.pid
- **start:** ./storectl.sh start [PROJECT_NAME]
- **license:** TBE

**Input/Output File Dependencies**
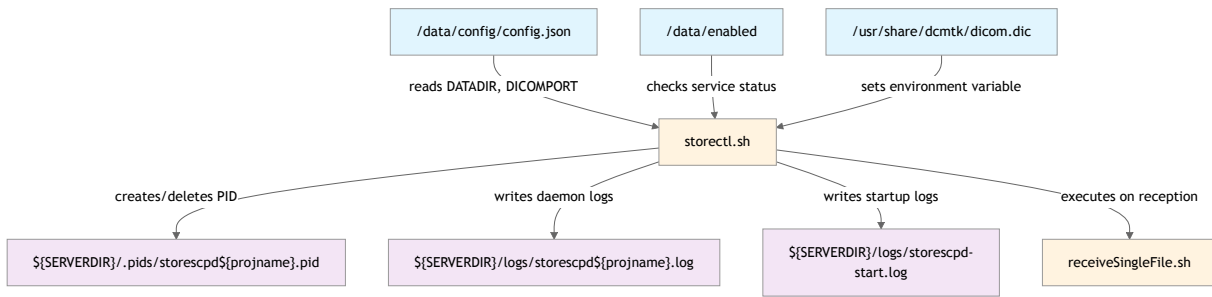


**File Descriptions:**

- `config.json` - Main configuration file containing DATADIR, DICOMPORT, and project settings
- `enabled` - Control file to enable/disable the service (first character: 0=disabled, 1=enabled)
- `storectl.sh` - Main control script for managing the DICOM storage daemon
- `storescpd.pid` - Process ID file for tracking the running daemon
- `storescpd.log` - Log file recording daemon activities and errors
- `processSingleFilePipe` - Named pipe for communicating file reception events to processing system

**Data Flow Dependencies**
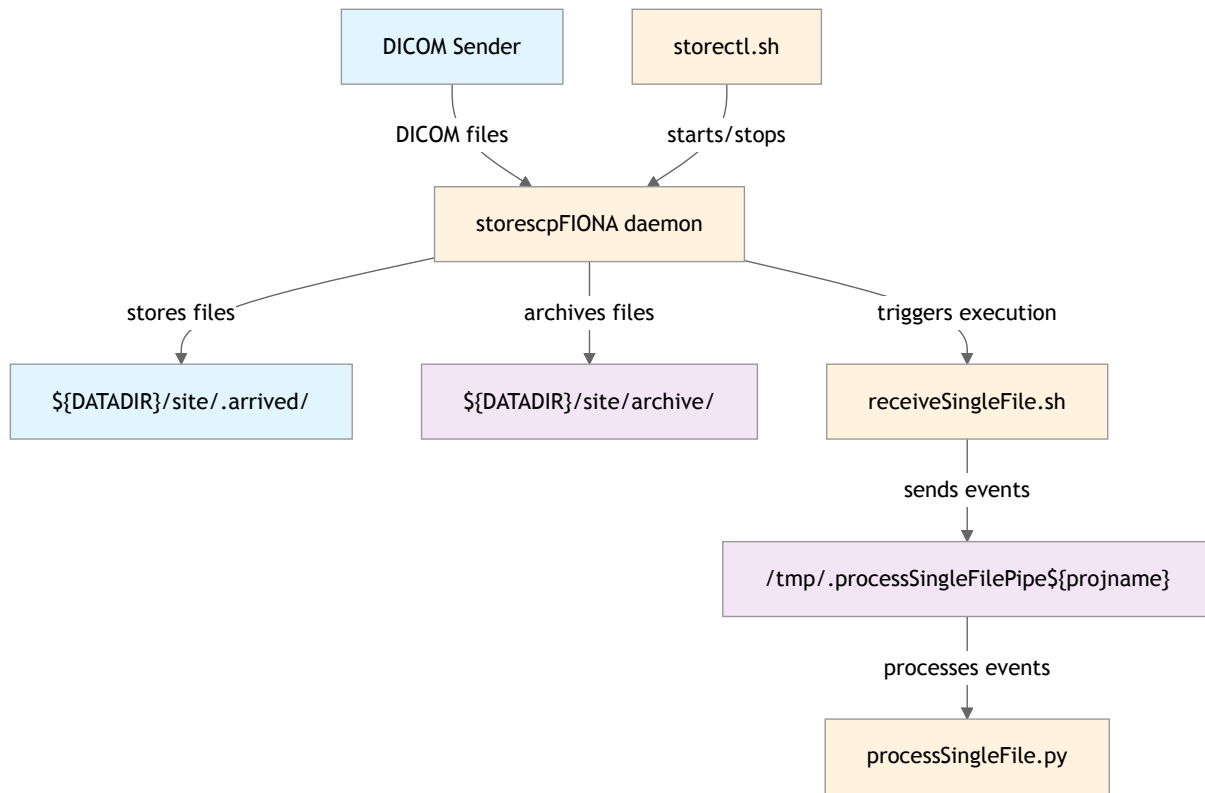
**File Descriptions**

1. Input Files:

- `/data/config/config.json` - Main configuration file containing data directories and DICOM ports
- `/data/enabled` - Optional control file to disable the service (first character "0" disables)
- `/usr/share/dcmtk/dicom.dic` - DICOM dictionary file for parsing DICOM data structures

2. Output Files:

- `${SERVERDIR}/.pids/storescpd${projname}.pid` - Process ID file for daemon management
- `${SERVERDIR}/logs/storescpd${projname}.log` - Main service log file for daemon output
- `${SERVERDIR}/logs/storescpd-start.log` - Startup log file for initialization messages
- `/tmp/.processSingleFilePipe${projname}` - Named pipe for inter-process communication

**Directories:**

- `${DATADIR}/site/archive` - DICOM file storage location
- `${DATADIR}/site/.arrived` - Temporary arrival directory
- `${SERVERDIR}/.pids/` - PID file storage
- `${SERVERDIR}/logs/` - Log file directory

### storectl.sh

filename: storescpd

purpose: start storescp server for processing user at boot time to receive data | Move files to project specific file system

This system service will fail if a control file /data/enabled exists and its first character is a "0".

(Hauke Bartsch)

## 3.3.2 processSingleFile3.py

`processSingleFile3.py` is a Python daemon process that monitors DICOM files, extracts header information, and creates Study/Series symbolic link structures. The script implements a generic daemon class and a specialized DICOM processing class that listens for incoming messages via named pipes, processes DICOM files, and organizes them into a structured directory hierarchy.

**The Main Dependences**

- **user:**
- **depends-on:**
- **log-file:** /../logs/processSingleFile[projname].log
- **pid-file:**
    - /../.pids/processSingleFile[projname].pid
    - /tmp/processSingleFile[projname].pid
- **start:** python processSingleFile3.py start [projname]
- **license:** Tbe

**The processing workflow:**

- Receives file paths via named pipe
- Reads DICOM files and extracts metadata
- Parses Siemens CSA headers for additional information
- Applies classification rules from classifyRules.json
- Creates organized directory structure with symbolic links
- Generates JSON metadata files for each series

**Input/Output File Dependencies Diagram**

config.json

classifyRules.json

named pipe

DICOM files

processSingleFile3.py

*.pid

*.log

*.json metadata

symbolic links

organized directories
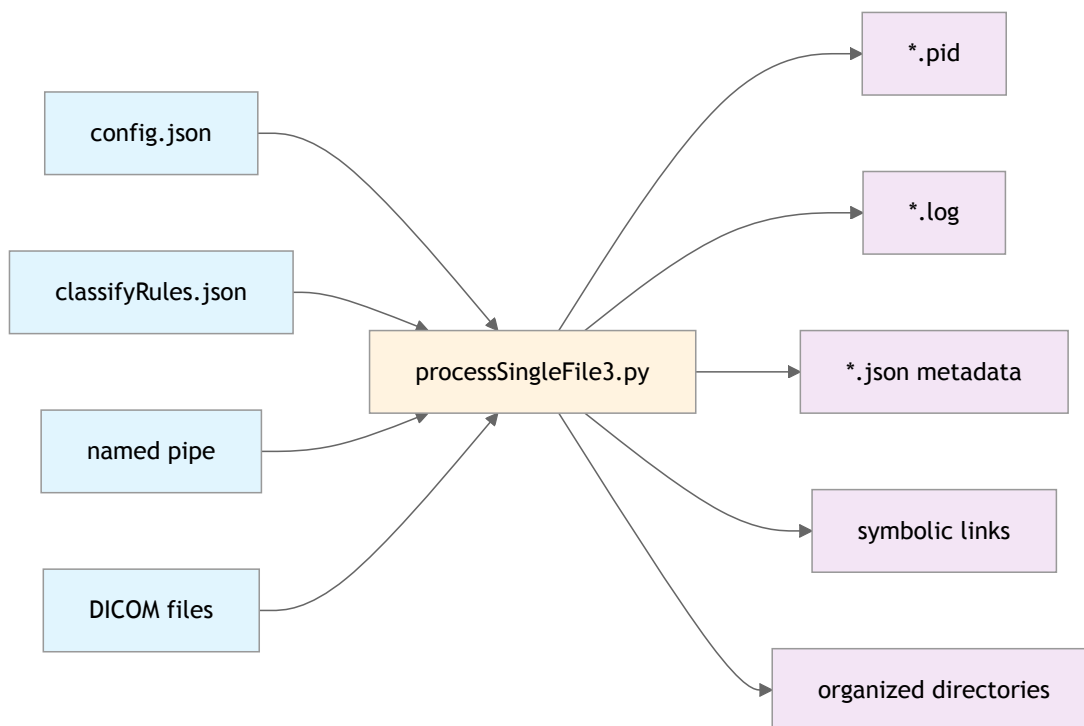
**File Description:**

1. Input Files:

- `config.json` - project configuration and directory paths
- `classifyRules.json` - DICOM series classification rules
- `named pipe` - daemon communication, receives file paths
- `DICOM files` - medical imaging files to be processed

2. Output Files:

- `*.pid` - daemon process ID
- `*.log` - operation and error logs
- `*.json metadata` - series metadata in JSON format
- `symbolic links` - links to original DICOM files
- `organized directories` - directory structure organized by patients/studies

**Data Flow Dependencies**
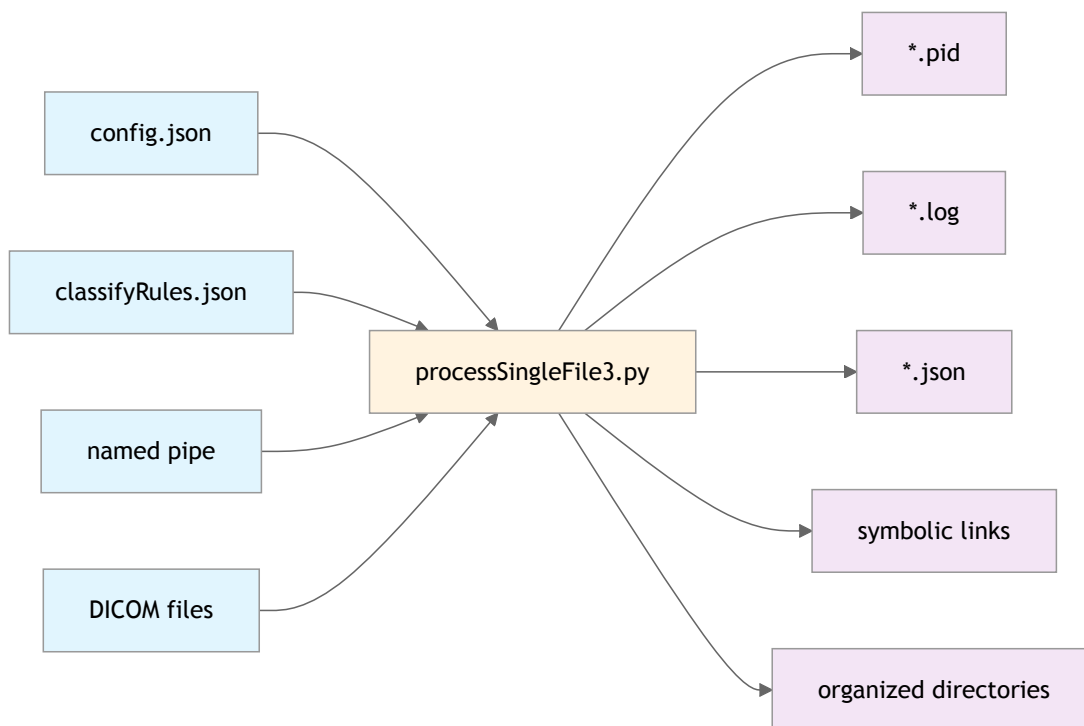
**Data Flow components:**

1. Input Files:

- `config.json` - system configuration file containing project settings and directory paths
- `classifyRules.json` - rule definitions for automatic DICOM series classification
- `named pipe` - inter-process communication channel receiving file processing commands
- `DICOM files` - medical imaging files in DICOM format containing patient data

2. Main Process:

- `processSingleFile3.py` - daemon process that orchestrates DICOM file processing and organization

3. Output Files:

- `*.pid` - process identifier files for daemon management and monitoring
- `*.log` - log files containing operational messages and error reports
- `*.json` - metadata files with extracted DICOM header information
- `symbolic links` - file system links organizing DICOM files by study structure
- `organized directories` - hierarchical directory structure arranged by patients and studies

**Directories:**

- `/data` - Default data directory
- `/data/site/.arrived` - Touch files for series arrival detection
- `/data/site/participants` - Patient-organized directory structure
- `/data/site/srs` - Structured reports directory
- `/data/site/raw` - Raw DICOM files organized by Study/Series
- `/data/site/temp` - Temporary files directory
- `../logs/` - Log files directory
- `../.pids/` - Process ID files directory

**processSingleFile3.py**

Create a daemon process that listens to send messages and reads a DICOM file, extracts the header information and creates a Study/Series symbolic link structure.

**The parser for the Siemens CSA header have been adapted from**
    https://scion.duhs.duke.edu/svn/vespa/tags/0_1_0/libduke_mr/util_dicom_siemens.py

### 3.3.3 detectStudyArrival

`detectStudyArrival.sh` monitors and processes incoming DICOM study arrivals in a medical imaging pipeline. The script checks for new study jobs created by receiveSingleFile.sh, processes them after a configurable delay, and triggers various quality control and compliance checks.

**The Main Dependences:**

- **user:** none
- **depends-on:**
    - receiveSingleFile.sh
    - anonymize.sh
    - /data/config/config.json
    - /tmp/.processSingleFilePipe
    - /var/www/html/applications/Assign/incoming.txt
- **log-file:**
    - ${SERVERDIR}/logs/detectStudyArrival.log
    - SERVERDIR/logs/detectStudyArrival{SERVERDIR}/logs/detectStudyArrival
    - SERVERDIR/logs/detectStudyArrival{projname}.log
- **pid-file:**
    - SERVERDIR/.pids/detectStudyArrival{SERVERDIR}/.pids/detectStudyArrival
    - SERVERDIR/.pids/detectStudyArrival{projname}.lock
- **start:**
    - ./detectStudyArrival.sh
    - ./detectStudyArrival.sh [PROJECT_NAME]
- **license:** TBE

**cron Configuration**

The script is designed to run via cron every 15 seconds to detect new arrivals and process studies for both ABCD and PCGC projects. Add these lines to crontab -e for 15-second monitoring intervals:

```
*/1 * * * * /data/code/bin/detectStudyArrival.sh
*/1 * * * * sleep 15; /data/code/bin/detectStudyArrival.sh
*/1 * * * * sleep 30; /data/code/bin/detectStudyArrival.sh
*/1 * * * * sleep 45; /data/code/bin/detectStudyArrival.sh
```

For PCGC project:

```
*/1 * * * * /data/code/bin/detectStudyArrival.sh PCGC
*/1 * * * * sleep 15; /data/code/bin/detectStudyArrival.sh PCGC
*/1 * * * * sleep 30; /data/code/bin/detectStudyArrival.sh PCGC
*/1 * * * * sleep 45; /data/code/bin/detectStudyArrival.sh PCGC
```

**Input/Output File Dependencies Diagram**
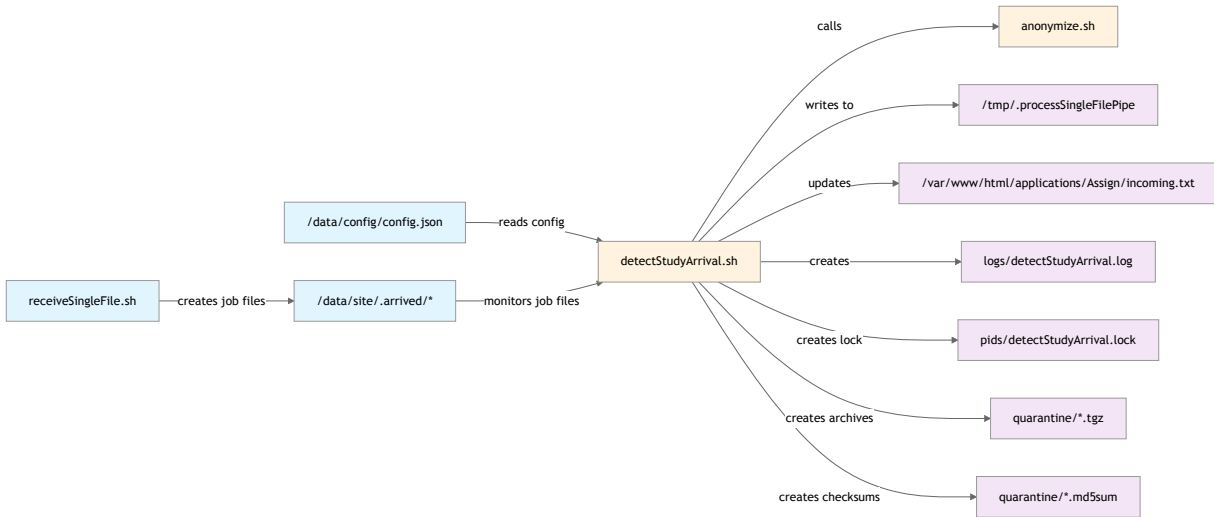
**File Descriptions:** | 1. Input Files:

- `/data/config/config.json` - Main configuration file with project settings
- `/data/site/.arrived/*` - Job files containing study arrival information
- `receiveSingleFile.sh` - Script creating study arrival job files

2. Output Files:

- `/tmp/.processSingleFilePipe` - Named pipe for queuing files to process
- `/var/www/html/applications/Assign/incoming.txt` - Web interface assignment list for studies
- `logs/detectStudyArrival.log` - Log file recording processing activities
- `pids/detectStudyArrival.lock` - Lock file preventing concurrent execution
- `quarantine/*.tgz` - Compressed archive files of processed data
- `quarantine/*.md5sum` - Checksum files for data integrity verification

3. Scripts:

- `detectStudyArrival.sh` - Main script monitoring DICOM study arrivals
- `anonymize.sh` - Script for anonymizing DICOM files

**Data Flow Dependencies**
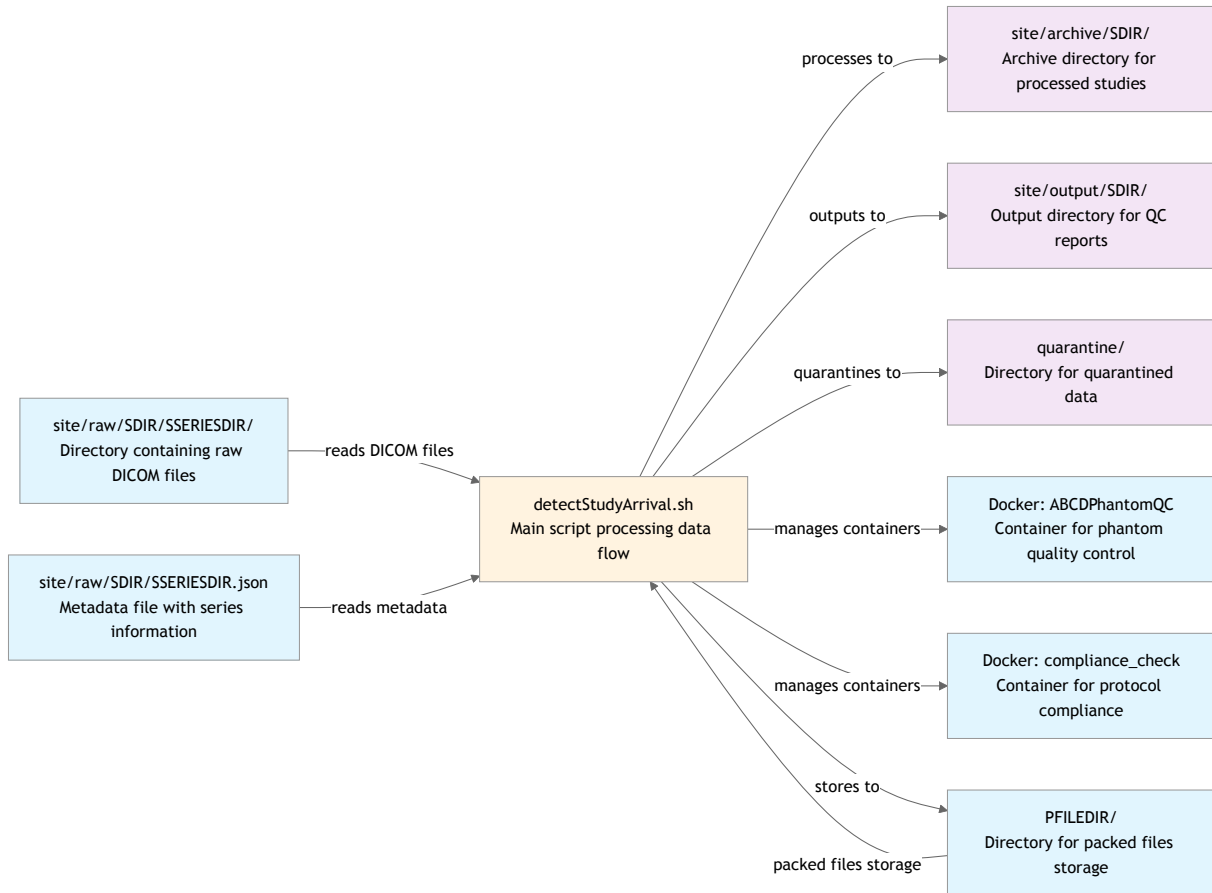
**Data Flow File Descriptions:**

1. Input Files:

- `site/raw/SDIR/SSERIESDIR/` - Directory containing raw DICOM files
- `site/raw/SDIR/SSERIESDIR.json` - Metadata file with series information
- `PFILEDIR/` - Directory for packed files storage

2. Output Files:

- `site/archive/SDIR/` - Archive directory for processed studies
- `site/output/SDIR/` - Output directory for QC reports
- `quarantine/` - Directory for quarantined data

3. Scripts:

- `detectStudyArrival.sh` - Main script processing data flow

4. Docker Containers:

- Docker: `ABCDPhantomQC` - Container for phantom quality control
- Docker: `compliance_check` - Container for protocol compliance

**Directories:**

- `/data/site/.arrived/` - Job arrival directory (ABCD)
- `/data[PROJECT]/site/.arrived/` - Project-specific arrival directories
- `${DATADIR}/site/raw/` - Raw DICOM storage
- `${DATADIR}/site/archive/` - Archived studies
- `${DATADIR}/site/output/` - Processing results
- `${DATADIR}/quarantine/` - Quarantine storage
- `${PFILEDIR}/` - Packed file directory
- `${SERVERDIR}/logs/` - Log file location
- `${SERVERDIR}/.pids/` - Lock file directory

## detectStudyArrival.sh

check the study job directory created by receiveSingleFile.sh

if the file is old enough process it using the information provided

Add this to "crontab -e" to check every 15 seconds if a new job arrived. `*/1 * * * * /data/code/bin/detectStudyArrival.sh`

```
*/1 * * * * sleep 30; /data/code/bin/detectStudyArrival.sh
```

```
*/1 * * * * sleep 15; /data/code/bin/detectStudyArrival.sh
```

```
*/1 * * * * sleep 45; /data/code/bin/detectStudyArrival.sh
```

```
*/1 * * * * /data/code/bin/detectStudyArrival.sh PCGC
```

```
*/1 * * * * sleep 30; /data/code/bin/detectStudyArrival.sh PCGC
```

```
*/1 * * * * sleep 15; /data/code/bin/detectStudyArrival.sh PCGC
```

```
*/1 * * * * sleep 45; /data/code/bin/detectStudyArrival.sh PCGC
```

### 3.3.4 anonymizeAndSend.py

There is no docstring.

2025.06.17 mk

### 3.3.5 sendFiles.sh

Example crontab entry that starts this script every 30 minutes

```
*/30 * * * * /usr/bin/nice -n 3 /var/www/html/server/bin/sendFiles.sh
```

Add the above line to your machine using:

```
> crontab -e
```

This script is supposed to send compressed data files for DICOM and k-space to the DAIC endpoint using sftp. All data in the /data/outbox directory will be send using local and DAIC md5sum files.

### 3.3.6 mppsctl.sh

filename: ppsscpfs

**purpose:**

- start DICOM multiple performed procedure steps server
- Keeps track of scans on the server:

```
*/1 * * * * /var/www/html/server/bin/mppsctl.sh start
```

(Hauke Bartsch)

### 3.3.7 parseAllPatients.sh

depends on output generated by getAllPatients2.sh

get list of optional findscu entries from http://dicom.nema.org/medical/dicom/current/output/html/part04.html#sect_C.6.1.1

### 3.3.8 process_tiff.sh

To be updated.

### 3.3.9 removeOldEntries.sh

Rremove any old entries from incoming.txt

### 3.3.10 runOneJob.sh

Run a single job from the workflow_joblist.jobs file. The file contains json code per line.

Need to run as user "processing" with flock.

**Example cron job:**

> **/usr/bin/flock -n /home/processing/.pids/Workflows_RunOneJob.pid**
> /var/www/html/applications/Workflows/php/runOneJob.sh >> /home/processing/logs/Workflows_RunOneJob.log
> 2>&1

### 3.3.11 getAllPatients2.sh

We can provide an argument to this program, the maximum number of days we would like to pull. In general we might get away with a very short period because new scans will come in as recent scans. But some test data migth be very old. So we should do one long run at night and short runs during the day.

As a second argument allow a specific project name. The whole things takes too long right now. Treat some project as special here.

### 3.3.12 heartbeat.sh

create a heart beat for the storescp

One way it can fail is if multiple associations are requested.

If the timeout happens the connection will be unusable afterwards.

Here we simply use echoscu to test the connection and if that fails we will kill a running storescp (hoping that monit will start it again).

In order to activate put this into the crontab of processing (every minute):

```
*/1 * * * * /usr/bin/nice -n 3 /var/www/html/server/bin/heartbeat.sh
```

### 3.3.13 moveFromScanner.sh

Called from cron-job, checks the study data in /dataPCGC/active-scans for things to pull from the scanner.

If all images are already found (only checks number of images) the series will not be pulled again. If all images of all series are present on the system the /dataPCGC/active-scans/<StudyInstanceUID> file is copied to the /dataPCGC/finished-scans/ folder.

This script will only run if there is no "/var/www/html/server/.pids/moveFromScannerPCGC.lock". This prevents multiple executions of this script in cases that a single processing steps takes more than 15 seconds.

This script will only run if there is no "0" in the second /dataPCGC/enabled bin (like in "101").

This script depends on the following dcmtk tools: dcm2dump, findscu, movescu.

Install using a cron-job every 15 seconds

```
*/1 * * * * /var/www/html/server/bin/moveFromScanner.sh
*/1 * * * * sleep 30; /var/www/html/server/bin/moveFromScanner.sh
*/1 * * * * sleep 15; /var/www/html/server/bin/moveFromScanner.sh
*/1 * * * * sleep 45; /var/www/html/server/bin/moveFromScanner.sh
```

```
*/1 * * * * /var/www/html/server/bin/moveFromScanner.sh PCGC
*/1 * * * * sleep 30; /var/www/html/server/bin/moveFromScanner.sh PCGC
*/1 * * * * sleep 15; /var/www/html/server/bin/moveFromScanner.sh PCGC
*/1 * * * * sleep 45; /var/www/html/server/bin/moveFromScanner.sh PCGC
```

### 3.3.14 clearExports.sh

find . -type d -maxdepth 1 -printf '%T+ "%p"n' | sort | less

delete oldest files until we have at least 20% free space on this partition

### 3.3.15 clearOldFiles.sh

find . -type d -maxdepth 1 -printf '%T+ "%p"n' | sort | less

delete oldest files until we have at least 20% free space on this partition

TODO: delete links in /data/site/participants/

TODO: delete links in /data/site/srs/

### 3.3.16 clearStaleLinks.sh

Delete links that do not point to real images in archive.

Just to be sure we have a clear /data/site/raw folder we should remove broken symbolic link and empty folders.

### 3.3.17 cron.sh

There is no docstring.

### 3.3.18 whatIsInIDS7.py

TODO: check if the number of study related series is correct (looks too large in Export app)

### 3.3.19 whatIsNotInIDS7.py

There is no docstring.

### 3.3.20 createTransferRequestProcessed.py

There is no docstring.

### 3.3.21 populateAutoID.py

Check all auto-id projects and create new transfer requests for each

### 3.3.22 populateIncoming.py

This program fills in the Study and Series information in the Incoming table in REDCap.

If a CouplingList entry exists its also adding a TransferRequest so that anonymizeAndSend can do its job.

TODO: support a new CouplingList entry even if there is already a TransferRequest done.

### 3.3.23 populateProjects.py

Each study that has been forwarded to PACS should appear in its own REDCap project.

We can get a list of all transferred studies from incomings transfers. We get a token for the project and add the entry - if it does not exist yet.

For informatino to appear in the Imaging instrument you need to set it up as a repeating instrument for "Event 1" (not a repeating event!).

TODO: Without calling for specific projects does not work anymore. We need to get a list of all imaging projects and run them project by project.

### 3.3.24 resendProject.py

check all transfer requests that have already been done

if the send date is before the request date send again

### 3.3.25 createTransferRequest.py

There is no docstring.

### 3.3.26 createZipFileCmd.php

This is a docstring.

# END USER / ADMIN / ARCHITECTURE (???)

**For:** us: mk & hb

## 4.1 * END USER (options) *

**For:** Doctors, researchers, medical personnel

**Should contain:**

- How to use the FIONA interface
- How to browse medical images
- How to export data
- Step-by-step instructions
- Interface screenshots
- FAQ for users

## 4.2 System Overview

The Fiona system is a comprehensive solution for managing DICOM medical images in a research environment. The system enables automatic reception, processing, anonymization, and transfer of imaging data between different PACS (Picture Archiving and Communication System) systems.

## 4.3 Core Functions for End Users

### 4.3.1 1.1 DICOM Image Reception and Processing

**Automatic Reception**: The system automatically receives DICOM images from various scanners and workstations

**Series Classification**: Images are automatically classified according to defined rules

**Multi-Modality Support**: MR, CT, US, SR (Structured Reports), PR (Presentation States)

### 4.3.2 1.2 Research Project Management

**Multi-Project Environment**: The system supports multiple independent research projects

**Data Routing**: Automatic routing of data to appropriate projects based on rules

**Auto-ID**: Automatic generation of participant identifiers for projects

### 4.3.3 1.3 Export and Archiving

**Multiple Export Formats:**

- Native DICOM

- NIFTI (for neuroimaging)

- PURE (directory structure by series description)

- Spectroscopy (spectroscopic data only)

**Anonymization**: Automatic removal of patient identifying data

**Secure Downloads**: Ability to create password-encrypted archives

### 4.3.4 1.4 REDCap Integration

**Data Synchronization**: Automatic synchronization with REDCap databases

**Transfer Requests**: Management of data transfer requests

**Metadata**: Storage and management of project metadata

## 4.4 Supported File Types

### 4.4.1 DICOM Images

- All standard DICOM modalities

- Multi-frame and enhanced DICOM objects

- Structured reports and presentation states

### 4.4.2 Export Formats

- **DICOM**: Original format with anonymization

- **NIFTI**: Neuroimaging format with JSON sidecars

- **PURE**: Organized directory structure

- **RAM-MS**: Specialized format for multiple sclerosis studies

- **Transpara**: Format for prostate imaging studies

## 4.5 User Interface Components

### 4.5.1 Exports Application

- Search and filter studies by project, participant, date

- Select export format and anonymization level

- Download encrypted archives

- Track export status and history

### 4.5.2 Assign Application

- View incoming studies awaiting assignment

- Manual assignment to research projects

- Coupling list management

- Study metadata review

### 4.5.3 Attach Application

- Upload whole slide imaging files

- Automatic conversion to DICOM format

- Metadata extraction and validation

- Integration with pathology databases

### 4.5.4 Workflows Application

- Launch containerized analysis workflows

- Monitor processing status

- Access results and outputs

- Integration with research pipelines

### 4.5.5 Support Contacts

Emails

## 4.6 * ADMIN (options) *

**For:** IT administrators, DevOps

**Should contain:**

- FIONA installation instructions

- Server configuration

- User management

- Monitoring and logs

- Backup and recovery

- Troubleshooting

- System requirements

## 4.7 System Architecture

The Fiona system consists of the following main components:

### 4.7.1 2.1 DICOM Servers

**storescp**: Server receiving DICOM data

**findscu/movescu**: Clients for searching and retrieving data from PACS

**echoscu**: Connection monitoring

### 4.7.2 2.2 Processing Components

**processSingleFile**: Daemon processing individual DICOM files

**detectStudyArrival**: Detection of completed image series reception

**anonymizeAndSend**: Data anonymization and transmission

### 4.7.3 2.3 Project Management

**populateIncoming**: Analysis of incoming data

**populateProjects**: REDCap project updates

**populateAutoID**: Automatic identifier generation

## 4.8 Installation and Configuration

### 4.8.1 2.1 System Requirements

**OS**: Linux (Ubuntu/Debian preferred)

**Python**: 3.x with libraries: pydicom, pycurl, json

**DCMTK**: DICOM tools

**Docker**: For containerized processes

**REDCap**: Research data management system

### 4.8.2 2.2 Directory Structure

```
/data/
├── config/
│   ├── config.json        # Main configuration
│   └── enabled            # Enable/disable flag
├── site/
│   ├── archive/           # DICOM archive (scp_<StudyInstanceUID>)
│   ├── raw/               # Raw data (StudyInstanceUID/SeriesInstanceUID)
│   ├── participants/      # Per-patient links
│   ├── srs/               # Structured Reports
│   └── .arrived/          # Arrival information for series
└── outbox/                # Data for external transmission
```

### 4.8.3 2.3 Configuration (config.json)

```
{
  "DATADIR": "/data",
  "DICOMPORT": "7840",
  "DICOMIP": "127.0.0.1",
```

```
    "DICOMAETITLE": "FIONA",
    "PROCESSING_USER": "processing",
    "SITES": {
      "PROJECT_NAME": {
        "DATADIR": "/dataPROJECT",
        "DICOMPORT": "7841"
      }
    }
}
```

## 4.9 Processes and Services

### 4.9.1 2.4 Main Daemons

1. **storescp** (storectl.sh): - DICOM listening port - File reception and storage - processSingleFile invocation

2. **processSingleFile** (processSingleFile3.py): - DICOM metadata processing - Image series classification - Directory structure creation

3. **detectStudyArrival** (detectStudyArrival.sh): - Series transfer completion detection - Post-processing initiation - Data archiving

### 4.9.2 2.5 Cron Jobs

```
# Basic processes (every minute)
*/1 * * * * /var/www/html/server/bin/storectl.sh start
*/1 * * * * /var/www/html/server/bin/heartbeat.sh
*/1 * * * * /var/www/html/server/bin/detectStudyArrival.sh

# Data processing (every 5 minutes)
*/5 * * * * /usr/bin/python3 /var/www/html/server/bin/populateIncoming.py
*/5 * * * * /usr/bin/python3 /var/www/html/server/bin/createTransferRequest.py
*/5 * * * * /usr/bin/python3 /var/www/html/server/bin/anonymizeAndSend.py

# Export (every 30 minutes)
*/30 * * * * /var/www/html/server/bin/sendFiles.sh

# Cleanup (daily)
0 2 * * * /var/www/html/server/bin/clearOldFiles.sh
0 3 * * * /var/www/html/server/bin/clearStaleLinks.sh
```

## 4.10 Monitoring and Troubleshooting

### 4.10.1 2.6 System Logs

- /var/www/html/server/logs/ - main system logs

- /home/processing/logs/ - processing user logs

- Monitoring through syslog

### 4.10.2 2.7 Common Problems

1. **Disk Space Issues**: clearOldFiles.sh, clearExports.sh

2. **Blocked Processes**: heartbeat.sh restarts services

3. **REDCap Problems**: Check tokens in tokens.json

4. **DICOM Connectivity**: echoscu for connection testing

## 4.11 Security

### 4.11.1 2.8 Data Anonymization

- Patient identifier removal

- Date modification (42-day shift for RAM-MS)

- DICOM tag export control

### 4.11.2 2.9 Access Control

- `processing` user for system processes

- `www-data` group for web interface

- 777 permissions for shared directories

## 4.12 Backup and Recovery

### 4.12.1 Backup Strategy

**Critical Data:**

- Archive directory (`/data/site/archive/`)

- Configuration files (`config.json`, `tokens.json`)

- REDCap databases

- Log files for audit trail

**Backup Schedule:**

- Daily: Configuration and logs

- Weekly: Full archive backup

- Monthly: Complete system backup

### 4.12.2 Recovery Procedures

1. **Service Recovery**: Use heartbeat.sh and systemctl

2. **Data Recovery**: Restore from archive backups

3. **Configuration Recovery**: Restore config files and restart services

4. **Database Recovery**: REDCap backup restoration

# 4.13 Performance Tuning

## 4.13.1 Optimization Settings

**Parallel Processing:**

- Multiple processSingleFile instances
- Chunked REDCap API calls
- Background export processing

**Storage Optimization:**

- Symbolic links instead of file copies
- Automatic cleanup of old data
- Compressed archives for export

**Network Optimization:**

- Connection pooling for DICOM operations
- Bandwidth limiting for external transfers
- Timeout management for long operations

# 4.14 Maintenance Procedures

## 4.14.1 Daily Tasks

- Monitor disk space usage
- Check service status
- Review error logs
- Verify REDCap connectivity

## 4.14.2 Weekly Tasks

- Clean up old export files
- Update routing rules if needed
- Backup configuration files
- Performance monitoring

## 4.14.3 Monthly Tasks

- Full system backup
- Update documentation
- Review user access permissions
- System security audit

# 4.15 * ARCHITECTURE (options) *

**For:** Developers, system architects

**Contains:**

- Source code documentation
- System components description
- APIs and interfaces
- Scripts and tools
- Data flow diagrams

## 4.15.1 Core Components

### Data Reception

- **storescpFIONA** - Custom DICOM SCP (Service Class Provider) server
- **storectl.sh** - Service controller for DICOM reception
- **Named pipes** - Communication mechanism between components

### Data Processing

- **processSingleFile3.py** - Main DICOM processing daemon
- **detectStudyArrival.sh** - Study arrival detection and workflow management
- **Classification system** - Rule-based study classification

### Data Management

- **File system organization** - Project-specific directory structures
- **Symbolic link management** - Study/Series organization
- **Data cleanup** - Automated file maintenance

### Data Transfer

- **anonymizeAndSend.py** - Data anonymization and transfer
- **sendFiles.sh** - Automated file transfer to research PACS
- **Transfer request system** - REDCap integration for transfer management
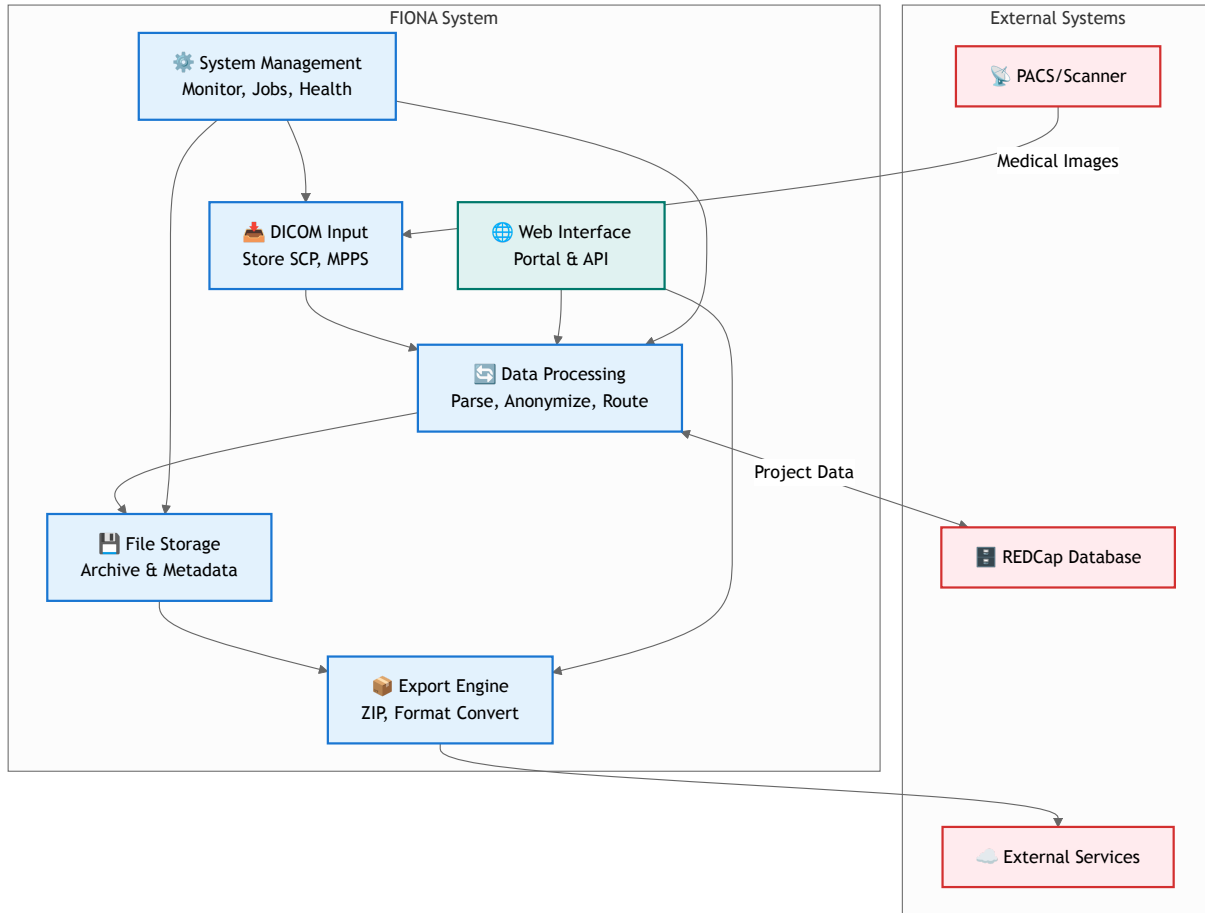
## 4.15.2 System Architecture

FIONA operates as a multi-layered system:

1. **Network Layer** - DICOM protocol handling
2. **Processing Layer** - Data classification and organization
3. **Storage Layer** - File system management
4. **Transfer Layer** - Data export and anonymization
5. **Management Layer** - Monitoring and control

General overwier (ver. 1)
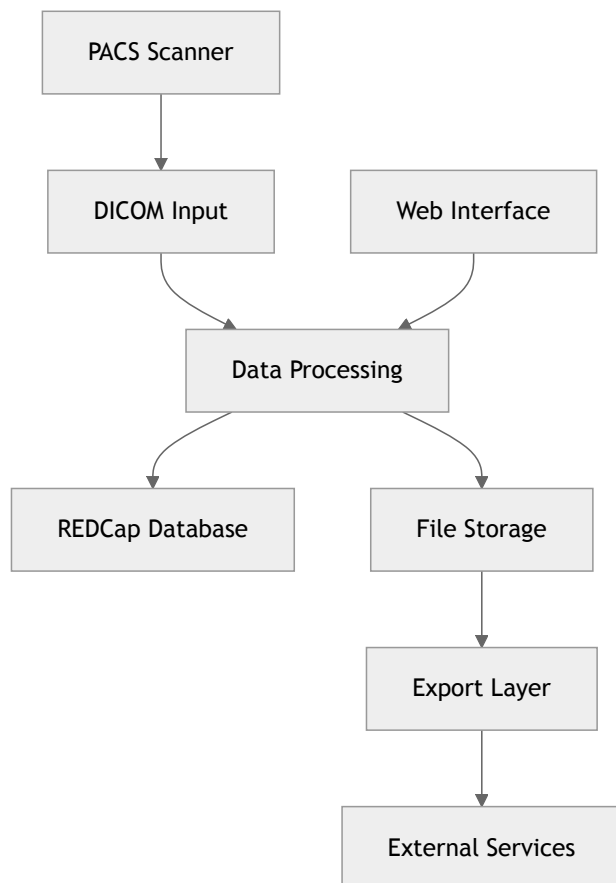
More detailed system overwier (ver. 2).

### 4.15.3 Key Features

- **Multi-project support** - Handles multiple research projects simultaneously

- **Automated workflows** - Minimal human intervention required

- **Data anonymization** - Compliant with research privacy requirements

- **Scalable design** - Can handle high-volume data processing

- **Monitoring and logging** - Comprehensive system monitoring

### 4.15.4 Technology Stack

- **Python** - Core processing logic

- **Bash** - System administration and automation

- **PHP** - Web interface components

- **DICOM toolkit** - Medical image handling

- **REDCap** - Transfer request management

- **Docker** - Containerized processing components

### 4.15.5 Deployment Model

FIONA is typically deployed as:

- **Single-server installation** - All components on one machine

- **Processing user account** - Dedicated system user for operations

- **Service-based architecture** - Daemon processes for continuous operation

- **Cron-based scheduling** - Automated task execution

Such an architecture ensures reliable, automated processing of medical image data while maintaining compliance with research and privacy requirements.

# FIVE

# CONTACT INFORMATION

- Website: https://fiona.ihelse.net

- Location: Haukeland University Hospital, Bergen, Norway

# INDICES AND TABLES

- genindex
- modindex
- search