NIH/NCI MIDI Phase 4: Final Report

Version: 2.0

Date: 12/06/2024

Authors: Laura Opsahl-Ong, Benjamin Kopchick, Kathryn Johnson, Theresa Do, Juergen Klenk, Dale Hawkins, Cheryl Corman

Prepared for: National Cancer Institute (NCI)

I. Overview

I.a. History of and Need for MIDI, Purpose of this Document

The Medical Imaging De-Identification (MIDI) project has been established by National Cancer Institute (NCI) to assess the feasibility of a tool to semi-automate the process of de-identifying medical images. The need for a tool such as MIDI has been discussed in the final report of the National Institutes of Health (NIH) Medical Imaging Working Group (MIWG), and it is one of its recommendations that NIH conduct a pilot to address this need. It is also discussed in further detail in the Objective section below.

The MIDI project has been performed in four main phases over the past four years. Phase 1 established the initial feasibility of a de-ID tool using the Google Cloud Platform's Healthcare API de-identification service. In Phases 1 and 2, the MIDI pipeline was set up and tested using a synthetic dataset created by UAMS. Based on the findings of Phases 1 and 2, Phase 3 was launched to implement a post-processing step in the MIDI pipeline in order to rectify many of the identified errors in Phases 1 and 2. It still utilized the same synthetic dataset from Phases 1 and 2. The results have been reported at the American Association for Cancer Research (AACR)[13]. Finally, Phase 4 moved from synthetic to real (full PHI/PII) imaging data, to conduct a realistic test of the pipeline. The resulting final tool, and its final results, as well as descriptions of the datasets, methods, and a discussion, are all included in this report.

This document has been created as part of Phase 4 of the MIDI project at the NCI. It is intended to provide a comprehensive overview of the activities completed during the MIDI Phase 4 Google Professional Services Organization (PSO) Engagement, and the status of the MIDI pipeline overall including its performance.

I.b Objective

With medical images now almost universally digitized and shared across health systems, proper data handling and patient privacy preservation are imperative. Handling and exchanging medical images present various challenges due to the Protected Health Information (PHI) and Personally Identifiable Information (PII) that can be embedded in the images either within the images themselves, as pixel data, or within textual metadata headers. Due to these challenges and the importance of ensuring patient privacy, concerted efforts have been made to develop and evaluate accurate and high throughput solutions for biomedical imaging de-identification.

To address the challenges posed by embedded PHI/PII, Digital Imaging and Communications in Medicine (DICOM) was established as an international standard for the communication and management of medical imaging information and related data. DICOM defines the formats for medical images that can be exchanged with the data and quality necessary for clinical use, including most radiology, cardiology imaging, and radiotherapy devices (e.g., X-ray, CT, MRI, and ultrasound). DICOM is also becoming more widely adopted within other medical domains, such as ophthalmology and dentistry. The thirty-year-old standard provides mechanisms to handle different writing systems, character sets, languages, and various addresses and legal name structures. It supports a multitude of biomedical imaging workflows, processes, and policies for different geographic regions, medical specialties, and local practices. Furthermore, the DICOM standard includes localization through procedural code sets and selective data element usage, enabling data from different geographic regions to meet the requirements of national or local health workflow policies.

Given the importance of these standards, especially in the context of public datasets, deidentification and adherence to DICOM standards are crucial to ensure patient privacy and data integrity. Public datasets often serve as valuable resources for research and development in medical imaging, making the proper handling and deidentification of PHI/PII essential to maintain confidentiality and comply with regulatory requirements. For instance, NCI's Imaging Data Commons (IDC), a cloud-based repository established in 2020, provides access to deidentified imaging data from eight different projects. The Cancer Imaging Archive (TCIA), an open-source imaging data resource established in 2011, is one of the main resources that populate IDC and contains tens of millions of biomedical images. With the ever-growing volume of imaging data, the standard process of removing sensitive information from an image, which historically involved manual inspection and editing of data, has become infeasible. Furthermore, applying these standardization practices to IDC and TCIA image data at such high throughputs is expensive and prone to error.

To make these processes more scalable, the development of reliable and robust automated image de-identification methodologies is essential. Approaches that automatically detect and remove PHI/PII would greatly benefit these already large and growing biomedical imaging databases. Furthermore, automated pipelines that integrate advanced machine learning (ML) and artificial intelligence (AI) can help improve de-identification accuracy and expedite the process, allowing image data to be shared more quickly. Ideally, the approach would also balance the removal of information that may contain sensitive details, to ensure privacy, with the retention of medical information that may be critical within a given research or clinical context.

De-identification of medical images in the DICOM format is a multi-step process that involves inspecting pixel data and metadata for PHI/PII where metadata elements values are either removed, kept, or cleaned based on the DICOM Security and System Management profiles.

MIDI offers an automated solution for de-identification of medical images that leverages Google Cloud Platform's (GCP) Healthcare Application Programming Interface (API), offering a secure, configurable, and scalable system for large datasets. To build the MIDI pipeline, the team had to define the processes, scripts, and configuration settings of the Google Healthcare API that could be used to perform de-identification of DICOM datasets in a manner compliant with the PS3.15 Attribute Confidentiality Profile. MIDI Phase 1 focused solely on native Healthcare API configurations but encountered several recurrent failure types. These errors stemmed from either unavailable options within the GCP platform or the application of algorithms not trained on specific PHI, such as abbreviated institution names. To address these recurrent error types, in Phase 3 we integrated customized post-processing scripts into the MIDI pipeline, resulting in a more robust and accurate approach. MIDI offers several key advancements over current state-of-the-art, including:

- Built within a secure cloud framework ensuring that PHI/PII is protected throughout the de-identification process.
- Novel cloud-based approach allows for scalable and efficient processing of large datasets.
- Implemented Cloud Functions from GCP to automate the pipeline without the need to manually trigger each step of the process.
- Alleviated limitations within the Healthcare API through use of regular expressions to alter specified data elements.
- Integrated custom post-processing scripts that address specific PHI-related issues not covered by standard algorithms leading to enhanced error handling.

Taken together, MIDI harnesses the established GCP Healthcare API platform and customized post-processing scripts to offer a more robust and accurate approach for biomedical image de-identification. Importantly, the approach integrates easily with ‘human-in-the-loop’ spot-checking, which could further improve the reliability of the de-identification process and enable progressive optimization of the workflow based on human feedback.

The Google Phase 4 PSO Engagement (MIDI Phase 4) was conducted to work in partnership with NCI, Deloitte, and key researchers from University of Arkansas for Medical Sciences (UAMS), Ellumen, and Leidos to assess and improve the performance of the MIDI pipeline for the de-identification of medical images, including the use of real (full PHI/PII) data.

For Phase 4, synthetic MIDI datasets and a real dataset were provided by UAMS that followed the DICOM standards. More detailed information about the datasets is provided below.

II. Background

Medical imaging data is an important tool for the diagnosis, treatment, and research of diseases. Sharing of imaging data among scientists can help accelerate research and discovery. Before images can be shared, however, all metadata and pixel data must be de-identified to comply with Health Insurance Portability and Accountability Act (HIPAA), other privacy regulations, and IRB requirements. Manual de-identification by humans is a repetitive and laborious process which does not scale with the rapid growth in biomedical data. Machine learning (ML) and deep learning (DL) algorithms, such as convolutional neural networks (CNNs) for images[1] or Recurrent Neural Networks (RNNs) for text[2], can be used to automate the process of removing sensitive information from biomedical data.

Publicly accessible image sharing repositories, such as The Cancer Imaging Archive (TCIA)[3] stand to benefit from automated de-identification to reduce the expense and improve the performance of maintaining them. TCIA includes cancer imaging data in Digital Imaging and Communications in Medicine (DICOM) format from various types of radiology modalities (CT, MRI, X-Ray, etc.). DICOM is the international standard format for acquiring, storing, processing, transferring, and visualizing medical imaging data in clinical environments[4]. However, for medical information to be distributed or shared for collaborative research purposes data and metadata must be reliably purged of identifiable information. Protected Health Information (PHI) and Personally Identifiable Information (PII) can be embedded in the pixel data of the images themselves as well as in the metadata (DICOM header). Maintainers of repositories, like TCIA, use the DICOM standard requirements for de-identification (specifically PS3.15 Appendix E) as a baseline for which elements are labeled as PHI/PII[5]. This standard provides instructions for selecting elements which are expected to contain PHI/PII, so that they can formulaically be removed (or replaced). A balance must be struck between removing information that is critical for interpreting the images within a certain research context, especially when the secondary re-use purpose is unanticipated[6]. Therefore, the DICOM standard describes a ‘profile with multiple options’ that range from being maximally restrictive (the ‘basic’ profile) to retaining more information, as well as providing for more complex “cleaning” of unstructured text, pixel data and other forms of data.

With an ever-growing volume of imaging data becoming available, a human manual approach to de-identification becomes infeasible, more expensive, and is always prone to error[7]. An automated and cloud-based system that employs ML/AI can help improve the de-identification accuracy, scale with datasets, and expedite the process allowing image data to be shared amongst researchers sooner. A recent study showed that fully customized systems can remove 97-99% of PHI/PII from text, while performance of off-the-shelf systems varied by dataset, with performance mostly above 90% sensitivity[8]. By contrast, human performance was evaluated at 81% in another study[9].

One approach for cloud-based de-identification of medical images is the Google Cloud Platform’s (GCP) Healthcare API[10]. Based on Google’s Data Loss Prevention (DLP)[11] service, it offers a configurable system that is scalable for large and growing datasets. It uses a combination of hardcoded rules and ML models including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for identifying PHI/PII in pixel data and DICOM header data. Post-processing scripts can also be included using cloud native functions for further improvements in accuracy.

The purpose of the MIDI Project was to develop a scalable, automated (or semi-automated), AI/ML-enabled and cloud-based image de-identification pipeline to provide service to large public repositories, such as the NCI Imaging Data Commons (IDC). This report describes Phase 4 of the MIDI Project that continues the previous work[12, 13] in improving the deidentification pipeline developed in Phases 1-3. Phase 4 tests primarily the MIDI pipeline developed in Phases 1-3 on a real PHI/PII dataset rather than just a synthetic dataset. During Phase 4, while setting up a secure PHI/PII computing environment, issues identified during retesting on the synthetic dataset were addressed. In addition, some modifications were made after testing on the test dataset used for the MIDI-B challenge[25]. Finally, some issues were only discovered during testing with the real PHI/PII dataset and were addressed as they were discovered. Figure 1 describes the flow of data for this phase of the project. There were two flows that MIDI was evaluated on. Flow 1 (red arrow) was using the Source dataset (see below for a description of all datasets) for the real PHI/PII dataset which came directly from the UAMS PACS. Flow 2 was using the Source dataset after being passed through the Clinical Trial Processor (CTP), configured with a UAMS-defined script that follows the DICOM PS3.15 profile, which performs an initial level of de-identification produced the “CTP” dataset. “CTP” is the dataset TCIA uses as input to its own more thorough human-drive de-identification and curation process (“TCIA Curated” or “UAMS” dataset). The TCIA curated “UAMS” dataset (in purple), assumed to conform to current best practices, was then used as the basis for determining how well MIDI performed in the comparison. In summary, MIDI was evaluated in two flows: directly working on the “Source” dataset, or indirectly working on the “CTP” dataset as input. In both cases, the TCIA curated “UAMS” dataset was used as the answer key to determine how well MIDI performed.

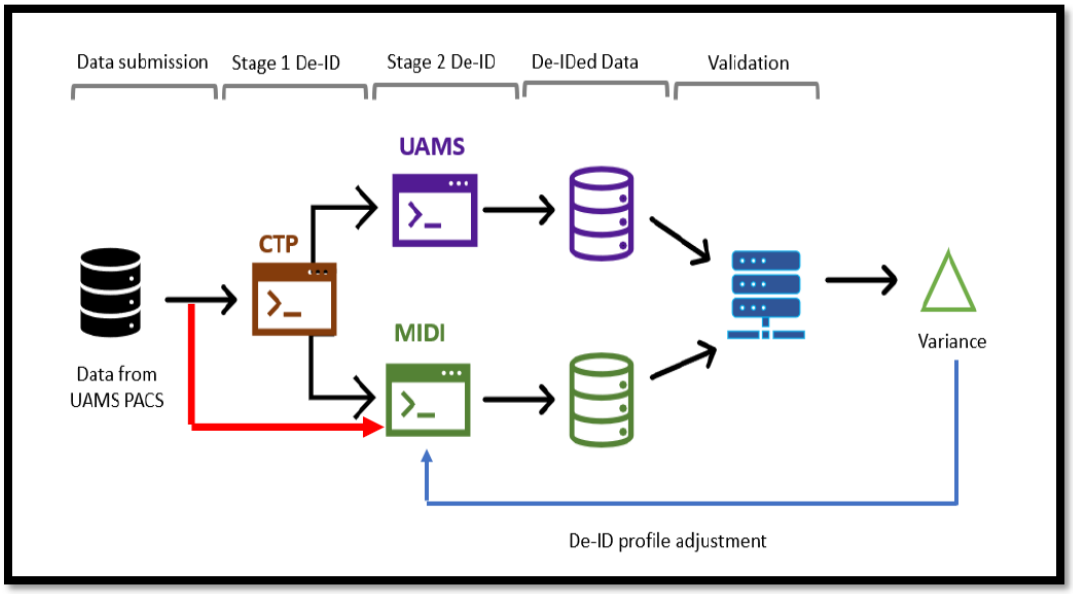


Figure 1. The evaluation flows for the Medical Image De-Identification (MIDI) project phase 4 with real PHI/PII.

II.a Timeline

Figure 2 shows the timeline of the MIDI project over the past 4 years.

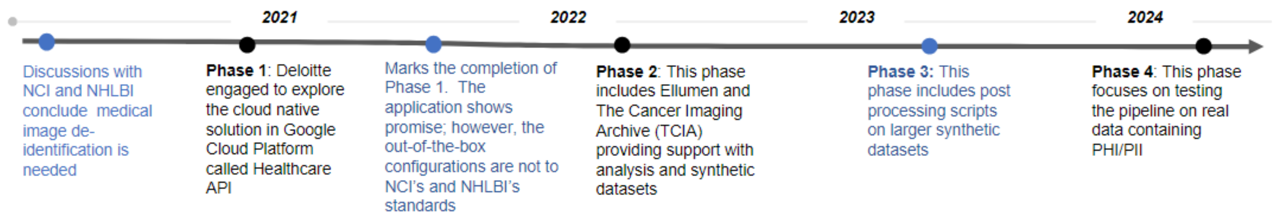

2021
2022
2023
2024
Discussions with NCI and NHLBI conclude medical image de-identification is needed
Phase 1: Deloitte engaged to explore the cloud native solution in Google Cloud Platform called Healthcare API
Marks the completion of Phase 1. The application shows promise; however, the out-of-the-box configurations are not to NCI's and NHLBI's standards
Phase 2: This phase includes Ellumen and The Cancer Imaging Archive (TCIA) providing support with analysis and synthetic datasets
Phase 3: This phase includes post processing scripts on larger synthetic datasets
Phase 4: This phase focuses on testing the pipeline on real data containing PHI/PII

Figure 2. Timeline of the Medical Image De-Identification (MIDI) project.

II.b Previous Phases of the MIDI Project

Previous phases of the MIDI project defined processes, scripts, and configuration settings of the Google Healthcare API that can be used to perform de-identification of DICOM datasets in a manner that conforms with the PS3.15 Attribute Confidentiality Profile.

The focus of the first two phases was to use only the configurations provided by the Healthcare API and not to include any additional outside scripts in order to keep the platform GCP native. Phase 1 focused on the development of a basic pipeline for customized deID of images, following TCIA best practices, and Phase 2 focused on further optimization based on findings during Phase 1.

The UAMS curated synthetic dataset was used for all runs during these phases[17]. Accuracy of the MIDI pipeline was measured against TCIA's standard tools and procedures for de-identification on a 21-patient dataset, consisting of a total of 1,693 individual images. Metrics included correct detection of all PHI/PII data and correct actions taken (e.g., remove, encrypt, or otherwise obscure). Throughput was also measured.

Throughput was measured at 22.0 images per second over 10 runs with images stored in a single region (us-east4). The MIDI pipeline's accuracy, calculated as number of correct actions taken per total number of actions required to be taken according to TCIA for DICOM headers and pixel data was 98.7%, accurately detecting dates, addresses, phone numbers, unique identifiers, names, and other common identifying information.

- The most common PII that the MIDI Phase 1/2 pipeline failed to remove were special cases that included uncommon non-western names (e.g., Bhavani) or names with symbols, dates in string data types that were mistaken for other IDs, patient IDs, and abbreviated institution names.
- Private Creator data elements consistently failed to be retained, rendering the corresponding private data elements unstable. These errors were due to a

misunderstanding of the role of private creator values, which cannot be changed.
- UIDs were correctly replaced.
- PII burnt-in to the pixel data was successfully detected and removed, with one false positive.

Further details on this phase can be found by reviewing our AACR abstract and presentation[12].

Phase 3 introduced additional pre- and post-processing scripts to add additional layers of de-identification to the DICOM files not natively supported by the Healthcare API de-identification. By introducing these new layers, we were able to reduce the total errors by 33% with the greatest reduction in errors being text not removed and tag not retained errors. See Table 1 for details of the MIDI Phases 2 and 3 pipeline results. Note that UIDs were not changed during these phases for easy comparison of input and output images and therefore the action to change UIDs is not shown here.

Table 1. Results based on de-identification actions comparing runs from 2nd phase and the 3rd phase.

Error Type	2nd Phase End Results	3rd Phase End Results
Date not Shifted	37	15
Text not Removed	446	259
Text not Retained	420	238
Tag not Retained	259	6
Text not Null	238	389
Total	**1355**	**907**

II.c DICOM Overview

What is DICOM

Digital Imaging and Communications in Medicine (DICOM)[14] is the most widely used and accepted international standard for the communication and management of medical imaging information and related data. It defines the formats for medical images that can be exchanged with the data and quality necessary for clinical use including almost every radiology, cardiology imaging, and radiotherapy device (X-ray, CT, MRI, ultrasound, etc.). It is also being used increasingly in devices in other enterprise medical domains such as digital pathology, dermatology, ophthalmology and dentistry.

DICOM is a world-wide standard that can be used in every locale. It provides mechanisms to handle data that support cultural requirements, such as different writing systems, character sets, languages, and structures for addresses and person names. It supports the variety of workflows, processes and policies used for biomedical imaging in different geographic regions, medical specialties and local practices. Localization to meet the requirements of national or local health and workflow policies can be done by including specifying code sets (e.g., procedure codes), or profiling data element usage (both specifying locally allowed values, and making elements that are optional in the Standard mandatory for local use).

DICOM Attributes

A DICOM object (such as an image) consists of pixel data and metadata, encoded as a list of “data elements”, each identified by a tag consisting of a pair of hexadecimal numbers. Some contain PII, such as Patient’s Name (0010,0010), and others do not, such as Number of Frames (0028,0008). The structure of the data element is as follows:

- **Tag** - A unique identifier for a Data Element composed of an ordered pair of numbers.
- **Value Representation (VR)** - Specifies the data type and format of the Value(s) contained in the Value Field of a Data Element.
- **Value Length** - The length of the Value Field of the Data Element.
- **Value** - A component of a Value Field. A Value Field may consist of one or more of these components.

More details on DICOM Data Structures and Encodings see DICOM PS3.5 2024e - Data Structures and Encoding.

II.d Data Sets

MIDI Phase 4 used multiple datasets to evaluate the success of the MIDI pipeline, both synthetic and, for the first time, a dataset with real PHI/PII. These datasets consisted of:

1. **MIDI Dataset:** An imaging dataset with synthetic PHI/PII prepared by UAMS.
    a. **MIDI 1.0 Dataset:** Synthetic dataset initially created for testing MIDI
    b. **MIDI 1.1 Dataset:** Synthetic dataset to improve on and expand MIDI 1.0 consisting of 6 groups.
        i. **MIDI-B Validation Dataset:** MIDI 1.1 Dataset Groups 1 and 2
        ii. **MIDI-B Test Dataset:** MIDI 1.1 Dataset Groups 3-5
2. **Source Dataset:** Imaging dataset that contains real PHI/PII
3. **CTP Dataset:** The source dataset that has been pre-identified by CTP
4. **UAMS Dataset:** The CTP dataset that was curated and de-identified by the UAMS curation team.

Prior phases of MIDI only used the MIDI 1.0 Dataset and MIDI 1.1 MIDI-B Validation Dataset. A history of the MIDI Datasets and their use through Phases 1-3 is provided below.

Synthetic MIDI Dataset

Table 2. The synthetic MIDI datasets that were generated over the course of the MIDI project. MIDI Dataset 1.0 was used in Phases 1 and 2 while groups of MIDI Dataset 1.1 were used in Phases 3 and 4 and the MIDI-B Challenge.

Name	Groups	Phases Used
MIDI 1.0 Dataset	1	1,2
MIDI 1.1 - MIDI-B Validation Dataset	1,2	3,4
MIDI 1.1 - MIDI-B Test Dataset	3,4,5	MIDI-B Challenge, 4 (through non-competitive runs of MIDI-B Challenge)

Synthetic Protected Health Information (PHI) and Personally Identifiable Information (PII) was generated and inserted into selected DICOM data elements and burned into pixel data to mimic typical clinical imaging exams. Information from the DICOM Standard as well as TCIA curation audit logs guided the insertion of synthetic values into both standard and private DICOM data elements and pixel data.

Three MIDI datasets were created during the duration of MIDI by UAMS. These dataset names, groups, and phases used are shown in Table 2. A description of the synthetic MIDI Datasets used in this project are documented[17] and a subset of 1.0 is made available[18]. The synthetic MIDI datasets were generated in multiple releases or versions.

MIDI Phases 1 and 2 used a subset of the 1.0 dataset, a 21-patient dataset. It was generated using a total of 1,693 images (CT, MRI, PET, and digital X-ray) selected from datasets published in TCIA. It follows the DICOM Standard and TCIA best practices for use in evaluating the performance of de-identification algorithms. The MIDI dataset generated for MIDI Phase 3 was created in a similar manner. It was split into six subsets, where groups 1 and 2 were used for validation of the MIDI pipeline and 3-5 were used in the MIDI-B challenge.

The full process of generating the synthetic datasets is summarized in Figure 3. Note that the subject and study counts apply to the MIDI 1.0 dataset, but the process applies to all of the synthetic MIDI Datasets.

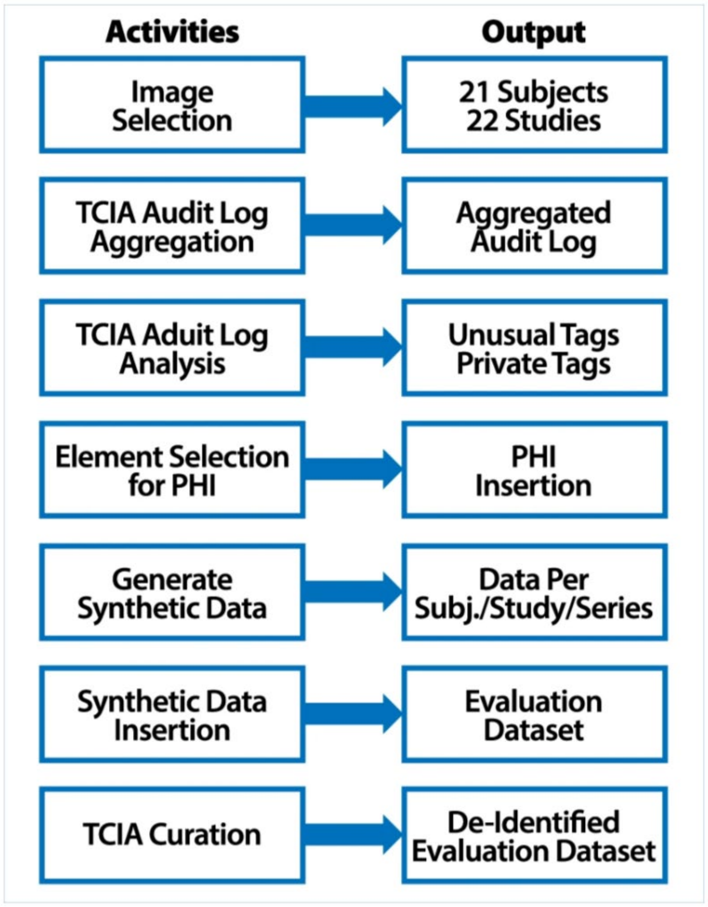


Figure 3. Creation of the synthetic dataset as completed by UAMS.

Selected imaging studies from TCIA were chosen to represent a broad cross-section of the current TCIA public collections. Table 4 breaks down the content of the synthetic evaluation set into the total number of patients, studies, series and images per modality, anatomy imaged by modality and manufacturers of imaging equipment used to collect the data. No images of heads were included to avoid subjects being identified by facial features.

For phase 3, the MIDI-B Validation Dataset used for testing contained 23,921 images consisting of 216 patients, 241 studies and 280 series. The counts for both MIDI-B Validation Dataset broken into groups 1 and 2 are below in Table 3. Also listed are the counts per imaging modality in Table 4.

Table 3. MIDI-B Validation Dataset counts used in Phase 3.

Group	Patients	Studies	Series	Instances
1	108	120	143	14662
2	108	121	137	9259

Total		216	241	280	23921

Table 4. MIDI-B Validation Dataset broken into Groups 1 and 2 counts by modality

Modality	Group	Patients	Studies	Series	Instances
CR	1	11	11	12	13
	2	11	12	12	13
CT	1	20	25	26	2217
	2	19	23	23	2189
DX	1	11	12	14	16
	2	11	13	13	18
MG	1	12	12	12	15
	2	13	13	13	16
MR	1	26	28	31	1723
	2	26	27	29	1788
PT	1	15	18	25	10655
	2	14	19	23	5069
SR	1	10	10	10	10
	2	11	12	12	12
US	1	12	13	13	13
	2	12	12	12	154

MIDI-B Test Dataset

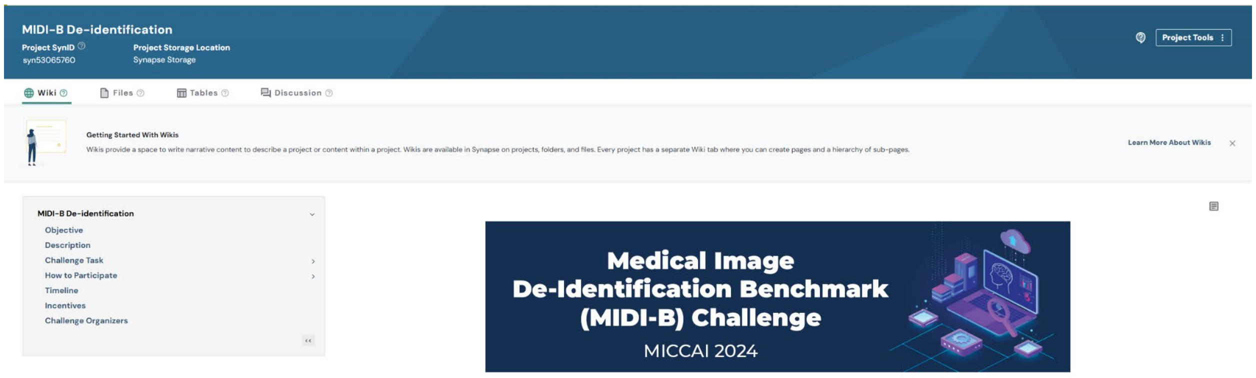
The MIDI-B Test Dataset, generated from groups 3, 4, and 5 of the synthetic MIDI 1.1 dataset, was an additional dataset that we tested. The dataset consisted of 322 patients and over 29,000 images.

Table 5. MIDI-B Test dataset counts by modality

Modality	Patients	Studies	Series	Instances
CR	65	73	75	78
CT	120	147	150	14517
DX	64	72	75	107
MG	74	74	75	90
MR	157	167	175	10828
PT	88	117	150	42187

SR	62	67	75	75
US	73	75	75	283

This dataset was used as the Test Phase of the MIDI-B Challenge[19] and more information can be found here: Synapse MIDI-B Challenge. Figure 4 is a screenshot of the MIDI-B challenge website as of 12/2024.


MIDI-B De-identification

Project SynID
syn53065760

Project Storage Location
Synapse Storage

Getting Started With Wikis

Wikis provide a space to write narrative content to describe a project or content within a project. Wikis are available in Synapse on projects, folders, and files. Every project has a separate Wiki tab where you can create pages and a hierarchy of sub-pages.

MIDI-B De-identification
- Objective
- Description
- Challenge Task
- How to Participate
- Timeline
- Incentives
- Challenge Organizers

Medical Image De-Identification Benchmark (MIDI-B) Challenge
MICCAI 2024

Figure 4. The MIDI-B Challenge front page as seen during the competition.

Source Dataset

The Source Dataset is the focus of this current phase. All previous phases concentrated on using the synthetic datasets described above, but this is the first time we used a real dataset containing real PHI/PII. This dataset is a subset of a Multiple Myeloma patient dataset that was chosen by UAMS due to its complexity.

The dataset consists of 393,899 DICOM instances which come from 302 patients, 799 studies, and 7542 series. It has representation for 18 different modalities as shown in Table 6. The only de-identification already performed on the source dataset is that which might have been performed by the submitting site prior to the application of the UAMS-supplied CTP process.

Table 6. Source Dataset Counts by Modality

Modality	Patients	Studies	Series	Instances
CR	129	146	636	657
CT	255	422	1155	214680
DX	3	4	4	4
ECG	1	1	2	2
KO	1	1	1	1
MG	2	2	7	7

MR	68	78	2481	54389
NM	52	55	116	240
OT	43	44	104	5461
PR	210	322	2340	2377
PT	203	203	405	114611
REG	1	1	1	1
RF	20	20	21	28
SEG	1	1	1	12
SR	26	26	28	30
US	44	47	47	436
XA	47	49	179	361
XD	13	13	14	351

CTP Dataset

The CTP Dataset is described in detail by UAMS' report and summarized here. It is a subset of the Source Dataset listed above, but passed through an initial phase of processing using the Clinical Trial Processor (CTP) as configured by UAMS with a script that follows the DICOM PS3.15 profile[5], and performs an initial level of de-identification[20]. This resulted in approximately 47,000 images to be quarantined. Of these files approximately 29,000 images were visually inspected and determined to not contain PHI/PII and were processed again by CTP, but without quarantining and added to the dataset. The remaining 18,000 images remained excluded from the CTP dataset. This resulted in the dataset described in Table 7.

Table 7. CTP Dataset Counts by Modality

Modality	Patients	Studies	Series	Instances
CR	126	143	626	634
CT	255	419	1076	214486
DX	3	4	4	4
ECG	1	1	2	2
KO	1	1	1	1
MG	2	2	7	7
MR	68	78	2457	54289
NM	25	25	65	189
OT	10	10	15	34
PR	210	322	2340	2377
PT	203	203	341	101826

REG	0	0	0	0
RF	9	9	9	11
SEG	0	0	0	0
SR	25	25	26	28
US	43	46	46	421
XA	44	46	176	358
XD	0	0	0	0

UAMS Dataset

The UAMS Dataset was the curated dataset used as the comparison for the MIDI pipeline results to be benchmarked against in Phase 4. It similarly started with the Source dataset, passed through CTP, and then was curated using TCIA best practices. The counts in comparison are very similar, however some images were specifically held out including all images with the modality of Presentation State (PR) as seen in Table 8.

Table 8. UAMS Curated Dataset by Modality

Modality	Patients	Studies	Series	Instances
CR	119	132	575	600
CT	255	419	1059	21441
DX	3	4	4	4
ECG	1	1	2	2
KO	1	1	1	1
MG	2	2	7	7
MR	68	78	2457	54289
NM	25	25	73	187
OT	1	1	5	5
PR	0	0	0	0
PT	203	203	338	101826
REG	0	0	0	0
RF	9	9	9	11
SEG	0	0	0	0
SR	24	24	25	25
US	3	3	3	52
XA	41	43	171	348
XD	0	0	0	0

II.e Google Isolator

To allow for the Source dataset to be uploaded to the Healthcare API and be accessed by the Deloitte team, Google required that Google Isolator[21] be implemented first. Isolator provides a layer of protection to the environment to ensure sensitive data is protected while being used in collaborations between multiple parties. It provided safeguards that included protecting data from accidentally being downloaded or being used on unapproved devices.

This was the first time Isolator was installed within the NCI CBIIT environment. It did take approximately 8 weeks to complete the setup as it required collaboration between Google, CBIIT, and CIT stakeholders based on how NCI's Cloud Two environment is set up.

III. Methods

De-identification of DICOM images is a multi-step process that involves inspecting metadata and pixel data for PHI/PII. The metadata data element values are either removed, kept, or cleaned. We start by using the DICOM PS3.15 Basic Application Level Confidentiality Profile using the Clean Descriptors Option. While DICOM PS3.15 provides multiple choices based on a datasets needed for de-identification, this was the agreed upon option for the MIDI pipeline. To meet this standard and to automate the process we configured a pipeline using GCP and the Healthcare API, and included additional processing scripts.

This work builds on previous MIDI phases to improve the de-identification results and attempt to fix problem areas identified above (Section: Previous Phases of the MIDI Project). Through small changes made to the algorithm (Section: De-Identification Scripts) and reviews of results we were able to consistently improve on the results. For example, identified limitations (Section: De-Identification Scripts) of the Healthcare API were fixed using Python scripts that use regular expressions to identify and remove PHI/PII in free text values. We also updated the pipeline to run using Cloud Functions that were parallelized to automate the pipeline without the need to manually trigger each step of the process. These steps and different scripts are described below.

The MIDI Phase 4 project made use of a public beta version of the Healthcare API rather than the general release version, in order to take advantage of specific features. These include the ability to specify tags to be kept, cleaned, or removed using Curl scripts. The additional pre-and post-processing scripts are written in Python using the open-source package Pydicom[22].

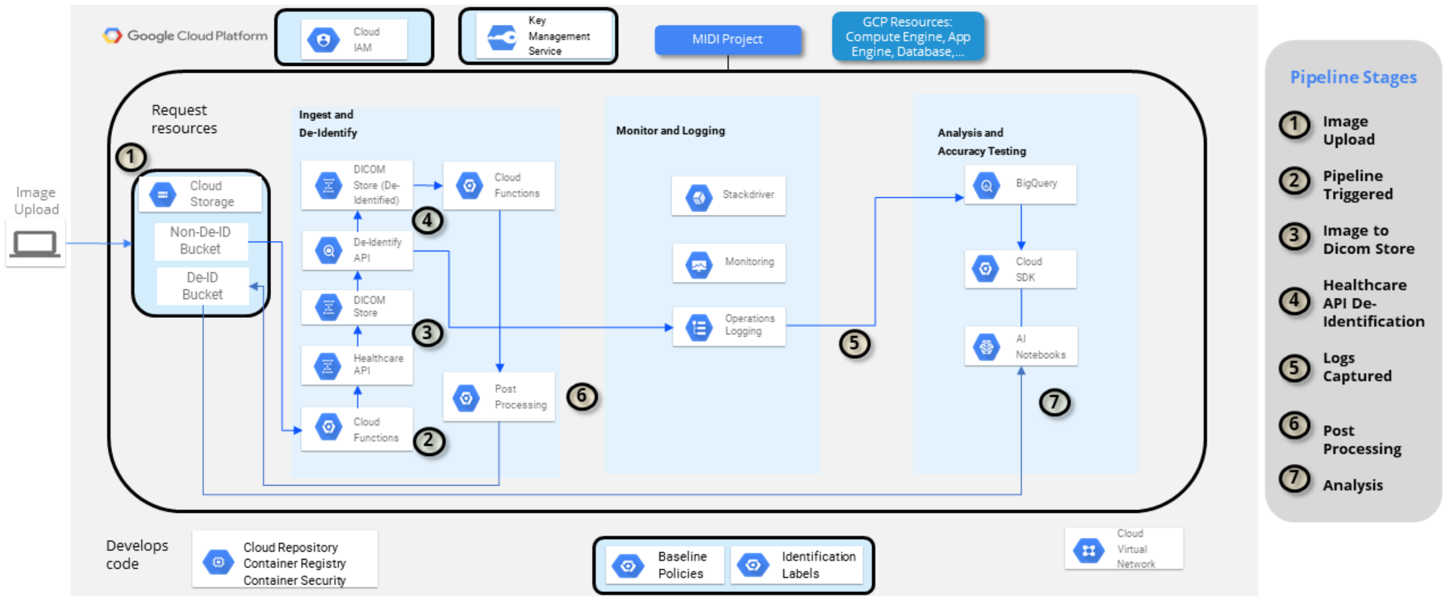


Figure 5. The MIDI pipeline architecture used in Phase 4.

For this phase, the infrastructure for the MIDI project was established on the NCI's Cloud 2 environment with the Project ID: nih-nci-cbiit-midi-dev2. The architecture of the de-identification pipeline includes the following stages which are valid for both single images and batch processing:

1) Load images into a Cloud Storage Bucket that has the appropriate firewall and security settings.
2) Trigger pipeline through the use of Cloud Functions.
3) Automated process transfers the image/images into an established DICOM Store using Cloud Functions that call the Google Healthcare API.
4) Run the Google Healthcare API de-identification service with the appropriate configuration flags. Google Healthcare API returns the de-identified image to a separately established De-Identification DICOM Store.
5) Logs are captured during the Healthcare API run, resulting in FHIR output that can be used to see all changes made to the DICOM images.
6) A second cloud function is initiated to run the post-processing, which returns the data to a specified Cloud Storage Bucket.
7) This step is part of the evaluation process of MIDI, and may be used in future evaluation/validation steps. Here we pull data from the Cloud Storage Bucket in a Virtual Machine environment and run comparison scripts prepared by TCIA and Ellumen for analysis. Additional analysis is also performed using Vertex AI Notebooks.

The pipeline is separated into two de-identification phases. One is to run the Healthcare API and the second is to run the post-processing scripts on the Healthcare API output. We currently keep these two processes separate because this project involved many updates to the post-processing scripts, for which the Healthcare API service did not need to be rerun. This improved our experimental efficiency and reduced cost.

II.b De-Identification Scripts

The MIDI pipeline is made up of two scripts, the GCP Healthcare API script that specifies the configurations of a GCP Healthcare API de-identification run, and a post-processing script which adds a further layer of de-identification to the DICOM files after they've gone through the GCP de-identification.
All scripts can be found on GitHub and instructions how to run them are at the end of this document in the Appendix.

Healthcare API Commands

GCP's Healthcare API allows for many configurations that we tested during this and the previous three phases. For this phase we are using the De-identify DICOM data using DicomTagConfig option which is, as of the date of this report, in the public beta. We will review the options we use here, however the full script can be found on GitHub,

- The first specified option is to keep certain data elements based on either their tag or their DICOM Value Representation (VR) (i.e., the "data type"). VRs decided to be retained which are not normally kept are included due to issues identified by purely using the Healthcare API to inspect and replace. This includes dates and date times. While the identification of dates by the Healthcare API performs reasonably well, there is an issue where there is a chance the randomly shifted date might shift by 0 (i.e., no shift at all). We work around this issue by having Google retain these, and doing date shifting in our post processing scripts instead.
- The second option is to remove data elements by their tag or VR. These have been determined through a systematic review and discussion amongst the MIDI team in tandem with using PS3.15 for data elements that should always be removed.
- The third option is to entirely replace values of certain data elements. This applies to data elements that must exist but can contain placeholder values.
- For all other values, including the pixel data, we use the Clean Text Tag option. This option uses GCP's DLP capabilities to identify and remove PHI/PII.
- We also activate the recursive tag option, which reviews all data elements nested within sequence items in the same manner as top-level data elements.We then use the regenerate option for Unique Identifier data elements (UIDs) that are meant to be regenerated (other than UIDs of well-known entities that are required to be retained, such as SOP Classes). UIDs are kept consistent across images by using a hash function.

- We then use the Operation Metadata feature, which writes all changes made to a FHIR store so that we can inspect the logic of the process, as well as create UID maps that are needed to match the before and after files for comparison. Changes recorded in the FHIR store are then read in BigQuery or Jupyter Notebooks using GCP's AI Platform.
- The final option is to select the profile type, which is the 'Deidentify Tag Contents' option, which follows the PS3.15 clean descriptor's de-identification profile.

There were eight iterations of the GCP script created and tested in this phase. The scripts and changes made for each script, as well as the decisions behind each change, are detailed in Table 6. All changes of how the GCP script treats specific DICOM data elements based on their tag or VR were based on consensus from the MIDI group. For example, if a data element with a specific tag resulted in many potential PHI/PII leak errors errors which could not be otherwise prevented and was also deemed not to be of research significance, the group decided it should be added to the remove list.

Post Processing Scripts

In MIDI Phases 1 and 2, de-identification was performed entirely using the GCP tool. However, we continued to run into certain errors that prompted us to create the post-processing step in Phase 3. These errors were the zero date shift (discussed above), as well as patient identifiers, some person initials, and location or institution acronyms that were not being identified and removed by the GCP tool (e.g.,: 'scan by RP' or 'performed at VCU'). Six versions of the Post-Processing script were made in Phase 3. They primarily focused on regenerating the Patient ID data element, using regular expressions to find and shift dates, and finding and removing Patient IDs, institution acronyms, person initials, and phone numbers in free text data elements.

There were eleven versions of the post-processing script created and tested in this phase. The primary focus of the work was on private data elements. Private data elements refers to a data element that is not defined by the DICOM standard, instead they allow manufacturers or users to add custom data not covered by existing data elements. Because private data elements require both the Private Creator and tag to be identified, and GCP identifies data elements by just using the tag, we ran into issues trying to keep or remove certain private data elements in the GCP script. For instance, if we wanted to automatically keep one private data element, but there was another private data element in the dataset with the same tag but a different Private Creator, the GCP script would keep the value. We moved our handling of private data elements over to post processing to prevent this issue from occurring. This and all other changes made for each version of the post processing script, along with the reasoning for each decision, are detailed in Table 8.

Private data element handling was first introduced in Post-Processing script v7. It checks whether a tag group number is odd (meaning it is a private data element), and then checks if that private data element exists in the TCIA keep list. Post processing script v8 made the improvement of correcting how the keep list was parsed through. Post processing script v9 made the fix of capitalizing letters in the DICOM tags before looking for a match in the TCIA keep list (which contains all capitalized letters). Post processing script v10 made the crucial change of basing the matching process on the Private Creator for the block. As discussed above, private data elements exist within a private data element block which is identified by the Private Creator data element for that block. A private data element cannot be correctly identified by tag alone, because multiple private data elements could exist in the same tag. Post-Processing script v11 then again fixes the capitalization issue, which occurred again when the major changes were made in the script for v10. Post-Processing script v12 made the final fix to private data element handling, which was to base the matching process not just on the most recent Private Creator element, but the Private Creator element that corresponds to the data element's tag.

Decisions Made

Table 9 - Decisions and reasoning made during this phase on changes to the de-identification scripts.

Decision Made	Reasoning	Script Version Implemented	Run Implemented For
First (failed) attempt to include private data element cleaning	The decision was made to automatically keep the private data elements determined as safe by TCIA and remove all other private data elements via the post-processing script. The TCIA keep list was provided by TCIA in the form of a csv and is a thorough list of private data elements they have determined to be both scientifically important and never contain PHI/PII. The csv was then uploaded into a Google cloud bucket where it could be accessed by the post processing script. Previously we had added private data elements to the GCP keep list when we encountered elements that should be kept in the synthetic MIDI dataset results. However, the GCP API does not allow users to accurately specify private data	post processing v7	Run 2024-07-12 (synthetic)

	elements. So a private data element in the keep list could end up keeping a completely different private data element that should actually be removed. Moving this feature over to post-processing allowed us to prevent this from happening.		
Expanded the phone number formats the script looks for and removes	Some phone number formats were still getting through	post processing v8	Run 2024-07-21 (synthetic)
The random seed for date shifting and patient ID generation were set with patient ID instead of series number.	Date shifting and patient ID generation was previously only consistent for the same series. So if a patient had multiple series within a study, the dates and patient ID generation was not consistent for that patient. Changing the random seed for date shifting and patient ID generation made it so these changes would be consistent for the same patient across multiple studies/series/instances.	post processing v8	Run 2024-07-21 (synthetic)
Slight fix to private data element handling	It took multiple attempts to get the private data element function in the post processing script working as intended. This was the first attempted fix. The details of changes made to private creator handling are described in the section above.	post processing v8	Run 2024-07-21 (synthetic)
Attempt to fix private data element handling.	It took multiple attempts to get the private data element function in the post processing script working as intended. This was the second attempted fix. The details of changes made to private creator handling are described	post processing v9	Run 2024-07-29 (synthetic)

	in the section above.		
Patient ID maps generated	The new version of the validation script required a before/after patient ID map as well as a UID map	post processing script v10 (but didn't do it right in automatic pipeline)	Run 2024-08-13 (synthetic)
Added in the data elements de-identification method, source application entity title, patient identity removed, and the de-identification entity title sequence.	New data elements were added in (primarily in group 2) that track changes made during the de-identification process. This was done to maintain DICOM compliance.	post processing script v10	Run 2024-08-13 (synthetic)
Major fix to how private data elements are cleaned	It took multiple attempts to get the private data element function in the post processing script working as intended. This was the third attempted fix. The details of changes made to private creator handling are described in the section above.	post processing script v10	Run 2024-08-13 (synthetic)
Moved some tags over to remove from reset so they won't be replaced by PLACEHOLDER. keeps the 5 tags in reset that are required to have a non-zero value	Many of the tags previously in the reset list actually belong in the remove list according to the DICOM standard. They were moved over in order to fix this issue.	script 8-13-24	Run 2024-08-13 (synthetic)
Used FHIR store change tracking in order to create a UID mapping. Switched to using de-identify dicomstore instead of de-identify dataset in order to accomplish this.	We stopped retaining the three main mapping UIDs (SOP Instance, Series, and Study UIDs) in order to test the fully de-identifying version of the pipeline. To create the UID mapping files, we had to track all UID changes with the FHIR store tracking feature. We also had to switch to using the de-identify dicomstore API	Script 8-01-24	Run 2024-08-13 (synthetic)

	because only that API contains the FHIR store tracking feature.		
Entered the new patient ID into the Patient Name data element.	Done since this is the standard procedure in DICOM de-identification	Post processing v 11	Run 2024-08-18 (synthetic)
Changed the time delta to +/-900 days (excluding 0)	Requested by group as better practice	Post processing v 11	Run 2024-08-18 (synthetic)
Replaced PLACEHOLDER with REMOVED for tags that are reset.	The GCP de-identification API replaces reset data elements with 'PLACEHOLDER', which sometimes goes over the character limit for those data elements. In order to solve that issue and make our results more similar to TCIA's, we replace these data elements with 'REMOVED' in the post-processing script.	Post processing v 11	Run 2024-08-18 (synthetic)
Removed 3 UID data elements from remove tag list, since they should be changed not removed	These UIDs had been added to the remove list in a previous phase of the project when UID shifting wasn't configured to work as consistently, and it was determined that these UIDs should be removed rather than remain unshifted. Now that UID shifting works correctly, these UIDs can be taken out of the remove list.	sc8-16-24	Run 2024-08-18 (synthetic)
Removes Patient Birth Date instead of shifting	Based on group consensus	Script 8-16-24	Run 2024-08-18 (synthetic)
Skip sequence data elements when looking for private elements.	There were some cases where private data elements that should be kept were in sequences. And because the sequence tag wasn't listed to be kept by TCIA, the entire sequence was removed	Post processing v12	Run 2024-08-26 (synthetic)

	(including the elements that should be kept). We changed the script to look through every private sequence and remove/keep data elements within the sequence according to the TCIA keep list.		
Final fix to private data element handling	It took multiple attempts to get the private data element function in the post processing script working as intended. This was the final fix. The details of changes made to private creator handling are described in the section above.	Post processing v12	run 2024-08-26 (synthetic)
Changed dateshift to -300 to -900 (excluding 0)	The group decided they preferred this date shifting window	Post processing v12	run 2024-08-26 (synthetic)
Used pre-created Patient's ID mapping and got rid of the old mapping tracker.	In order to change Patient IDs exactly the same as TCIA (so we could minimize discrepancies in the comparison results), we began using a pre-created patient ID mapping file as a guide for how to shift patient IDs in the post-processing script.	Post processing v12	run 2024-08-26 (synthetic)
Changed the way that files are uploaded to GCP in a way that updates the header info.	According to the DICOM standard, the group length data element in the PS3.10 meta information header needs to be updated with the new length when changes to the header data are made. We changed the way we upload the files back to GCP storage after post-processing using pydicom so that pydicom automatically updates the group length.	Post processing v12	run 2024-08-26 (synthetic)
Removed the Series Number data element	Determined by group consensus	sc8-26-24	run 2024-08-26 and run 2024-

(0020,0011) from remove list and added to keep list			08-30 (synthetic)
Removed Patient's Birth Date in post processing.	For some reason removing it wasn't working in GCP.	Post processing v13	run 2024-08-30 (synthetic)
Added implementation version name as gcpv1beta1.	Implementation Version Name, according to the DICOM standard, needs to be updated with the version of the tool used for de-identification.	Post processing v13	run 2024-08-30 (synthetic)
Updated private tag finder to use capital letters.	Previously, private data element tags in the TCIA keep list contained letters that were capitalized, while the letters in tags within pydicom files were lowercase. This led to private data elements that should be kept not being identified. We changed the script to account for this.	Post processing v13	run 2024-08-30 (synthetic)
Entered the new Patient's ID into Clinical Trial Subject ID and Clinical Trial Reading (if it exists).	This is the TCIA standard for de-identification.	Post processing v13	run 2024-08-30 (synthetic)
Replaced verifying observer name and person name with REMOVED.	Group consensus decision made to match results closer to TCIA output	Post processing v14	run 2024-09-05 (synthetic)
Clinical Trial Subject ID is replaced with the new patient ID.	This is the TCIA best practice for de-identification.	Post processing v14	run 2024-09-05 (synthetic)
Removed Retain Device Identity from de-identification entity title sequence.	We determined that we were not using the Retain Device Identity option.	Post processing v14	run 2024-09-05 (synthetic)
Added Clinical Trial Protocol ID, Clinical Trial Subject ID, and	Group consensus decision made to match MIDI output closer to TCIA output	sc9-05-24	run 2024-09-05 (synthetic)

Clinical Trial Subject Reading ID to reset tag from remove list so that they will be replaced with 'REMOVED'			
Tailored the script to the CTP run. Removed date shifting. Removed dates found not in DA or DT. Removed patient ID changing.	The script for the CTP run had to be altered because the CTP dataset had already been mostly de-identified by the CTP, so things like date shifting and changing patient ID were redundant.	Post processing v15	run 2024-09-25 (ctp)
Puts REMOVED into Presentation Creator's Name	Group consensus decision made to match MIDI output closer to TCIA output	Post processing v15	run 2024-09-25 (ctp)
Changed Clinical Trial Subject ID and Clinical Trial Subject Reading ID to equal str(new patient id) instead of the new patient id in integer form	When we first added patient ID into these data elements in the previous post processing script, the integer version of the new patient ID instead of the string version was added, resulting in formatting mistakes.	Post processing v15	run 2024-09-25 (ctp)
Tailored for CTP run. No UID changes or fhir store tracker.	The script for the CTP run had to be altered because the CTP dataset had already been mostly de-identified by the CTP, so things like UID shifting were redundant.	sc9_13_24	run 2024-09-25 (ctp)
Re-tailored post processing script for non-CTP run	Added back in features such as patient ID changing and data shifting.	Post processing v16	run 2024-10-08 (source)
No longer put REMOVED in Presentation Creator's Name	Decision made to match MIDI output closer to TCIA output.	Post processing v16	run 2024-10-08 (source)
Introduced parallel processing.	The post-processing run on the CTP data took about a week because the dataset is so large, so we updated the	Post processing v16	run 2024-10-08 (source)

	post-processing script to involve parallel processing so that the run could be done in an hour or two.		
Added all of the Private Creator tags found in TCIA's keep list to the GCP keep list	Done so that Private Creators are never removed, which caused the private data element function in the post processing script to stop working when it happened.	sc10-04-24	run 2024-10-08 (source)
Removed Curve Data and Overlay Data data elements.	Done according to DICOM standard.	Post processing v17	Run 2024-10-25 (source)
Set Media Storage SOP Instance UID to new SOP Instance and UIDvalue	Done according to DICOM standard.	Post processing v17	Run 2024-10-25 (source)
Added Secondary Capture Device ID and Hardcopy Creation Device ID and Referenced File ID to remove list	Group consensus decision.	sc10-25-24	Run 2024-10-25 (source)

II.c Evaluation

Evaluation of the MIDI pipeline was conducted with three runs. The first run used the synthetic MIDI 1.1 Datasets, the second used the UAMS Curated Dataset, and the third used the Source Dataset. Results of the MIDI pipeline from each run were then compared to desired results as described below. Multiple de-identification runs were made throughout the MIDI Phase 4 PSO Engagement. This allowed the project team to analyze the results, make modifications to the methods (in particular, the post-processing script) outlined above, and improve the accuracy of the results.

MIDI 1.1 Dataset Run

The MIDI 1.1 Dataset Run utilized the curated synthetic MIDI Dataset, which included an answer key and a validation script to compare the MIDI pipeline results to an answer key.

The validation script[23] written in Python is provided for comparing the answer key to a de-identified version of the synthetic data. The script utilizes answer key files generated during the creation of the synthetic datasets along with mapping files for both UIDs and patient IDs, which map pre-curated IDs to post-curated IDs, to validate all data elements for all DICOM images in a curated dataset.

We directly viewed the total number of errors based on actions that were required to be taken. The errors were grouped as:

1. **Text Removed -** where text which was deemed to be PHI/PII was not removed
2. **Text Retained -** where text which was deemed not to be PHI/PII was removed
3. **Tag Retained -** where tags were failed to be kept, usually necessary for DICOM compliance
4. **Text Not Null -** where text was blank, usually necessary for DICOM compliance
5. **UID Changed -** where UIDs were not properly altered
6. **Date Shifted -** where dates were failed to be shifted
7. **Pixels Hidden -** where burnt-in pixels with PHI/PII were failed to be hidden
8. **Pixels Retained –** where pixels not containing PHI/PII were obscured
9. **UID consistency –** UIDs were changed but not consistently for a given patient
10. **Patient ID consistency –** Patient IDs were changed, but not changed consistently

UAMS Curated Dataset Run

The UAMS Curated Dataset Run utilized the UAMS Curated Dataset, which included a comparison script. This script did a direct comparison of the differences between the MIDI output and the UAMS Curated Dataset. These differences were recorded in excel files for analysis which is discussed in the below Results & Discussion section.

Source Dataset Run

The Source Dataset Run utilized the Source Dataset containing real PHI/PII, which did the same comparison as the UAMS Curated Dataset runs. The comparison script was used to compare the outputs for the Source and CTP datasets. These scripts perform a one-to-one comparison with the files in the TCIA Curated Dataset, which was used as the answer key, and then evaluated by the team to identify errors in the MIDI pipeline output. The results are discussed in the below section.

IV. Results & Discussion

The sections below review the results of our most recent runs on the datasets for both the Pixel Data and the Header Data.

IV.a Pixel Data

The tables below show a comparison of the error types identified during our most recent runs for the pixel data. They are broken up by dataset.

Run using MIDI-B Validation Dataset comparing MIDI Pipeline output with Answer Key

The results on the pixel data can be seen in Table 10. Of the 23,933 instances, there were ten images that required PHI/PII to be removed and all were correctly completed. There were two additional images that contained false positives and are displayed below. This results in an error rate of 0.008% for the pixel data on the MIDI-B Validation Dataset.

Table 10. Pixel actions taken on the MIDI-B Validation Dataset by the MIDI pipeline

Pixel Actions	Count
Correct PHI/PII Removal	10
False Positive PHI/PII Removal	2
False Negative PHI/PII Removal	0
No Action Required and No Action Taken	23921

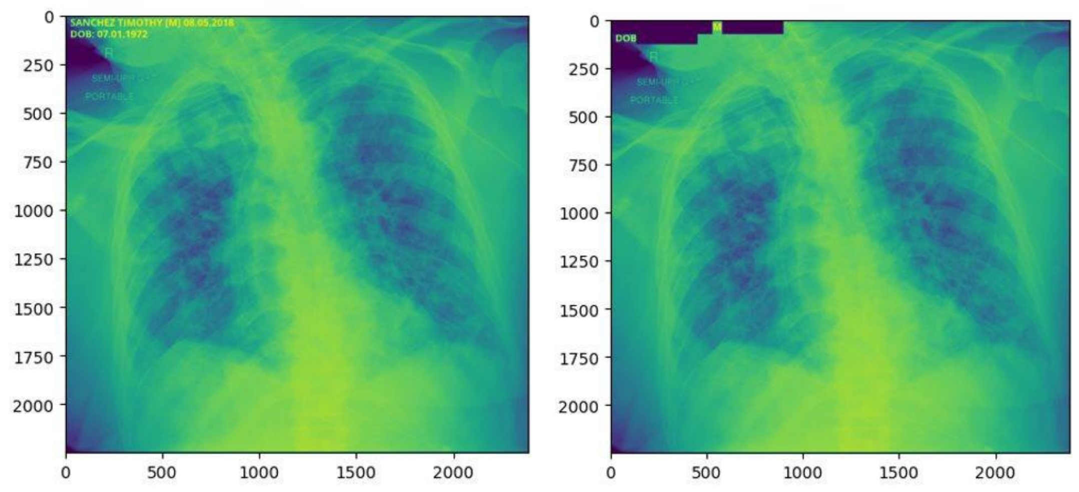


Figure 6. Example of PHI/PII being correctly removed.

Figure 6 is an example of PHI/PII being correctly removed with black bounding boxes replacing the sensitive text.

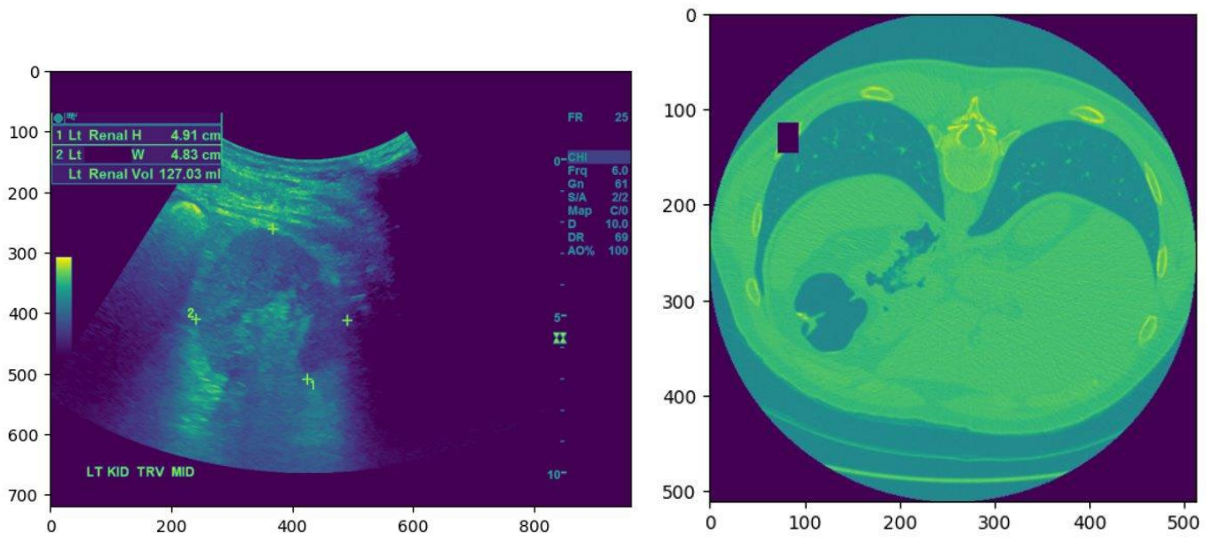


Figure 7. Example of false positives, where bounding boxes were incorrectly placed.

In Figure 7 there are the two instances of false positive PHI/PII removal. On the left image, text is removed on the right-hand side that does not need to be removed. On the right image, notice part of the body is incorrectly being removed. This is a common issue that is also seen in the source dataset.

There were 20 additional files flagged by the validation script for containing false positive pixel removals. However, all 20 of these files contained images within 3d multi-frame arrays, instead of the typical 2d single-frame pixel arrays seen in the rest of the dataset. When we examined the contents of the pixels post de-identification, no changes were found. This leads us to believe that there was an issue with how the validation script examined 3d multi-frame arrays, causing false flags on 20 files. An example of one of the file's images can be seen in Figure 8.

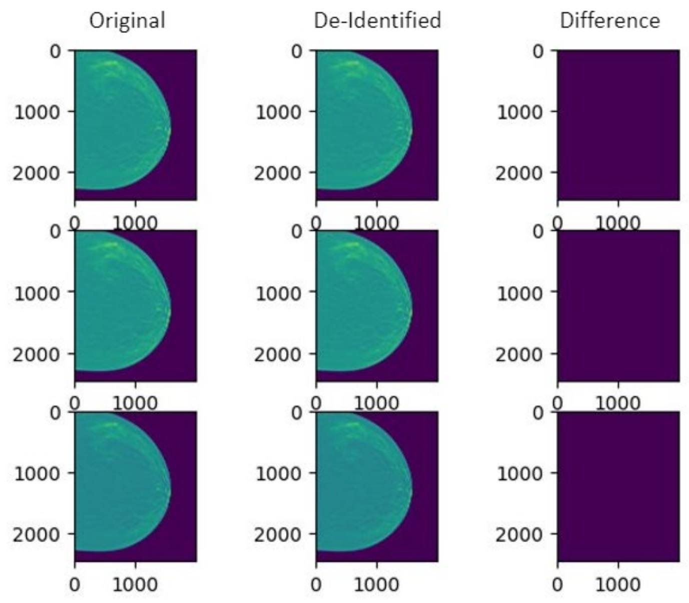


Figure 8. Example of 3D multi-frame array that was incorrectly flagged by the validation script, however MIDI pipeline correctly did not incorrectly remove any pixels.

Run using CTP Dataset comparing MIDI Pipeline output with UAMS Curated Dataset

Here we show the discrepancies between the MIDI pipeline being run on the CTP dataset compared with the UAMS Curated datasets. Table 11 shows these discrepancies based on where the MIDI pipeline incorrectly removed a part of the image which would severely impact the image analysis such as putting a black box over an organ as well as minor false positive image removal such as removing part of a table or corner of an image deemed inconsequential to human interpretation of the image (However, it should be noted that removing pixels from any part of the image, even if it may not impact human analysis of the medical information contained, can still severely impact the performance of AI algorithms and undermine their generalizability). We also identified cases where text was removed, but it was not sensitive, cases where the MIDI pipeline failed to remove acronyms that TCIA removes by their best practices, and then the correct PHI/PII removal.

Table 12 identifies the instances where there were identity leaks due to patient IDs being burned in but not successfully removed by the MIDI pipeline (i.e., false negatives).

Table 11 - Pixel Data Discrepancies between MIDI pipeline output and UAMS Curated Datasets. False positive image removal includes any false positives where vital information is removed. Minor false positives include pixels removed on boundaries or tables. Major False Positive Text Removal is any text that was unnecessarily removed in the pixel data. TCIA

Acronym Removal includes acronyms failed to be removed that are by TCIA. PHI/PII removal are failures to remove PHI/PII.

MIDI Pipeline vs UAMS Curated Dataset Discrepancies	
Discrepancy Type	**Count**
MIDI Pipeline False Positive Image Removal	25
MIDI Pipeline Minor False Positive Image Removal	7
MIDI Pipeline Major False Positive Text Removal	31
UAMS Acronym Removal	52
PHI/PII Removal	53

Table 12 - Pixel Data PHI leaks analysis using MIDI pipeline compared to UAMS Curated Datasets

MIDI Pipeline vs UAMS Curated Dataset PHI/PII Leaks	
PHI Leak Type	**Count**
None	105
Patient ID	52

Overall, the UAMS and MIDI pipelines concurred on most of the files that were in UAMS's Curated dataset of 371,793 instances. In total, 63 images contained false positives, a rate of 0.015%. Of the remaining PHI leaks, all were either Acronyms being failed to be removed or Patient IDs being failed to be removed. The Patient IDs are a special case where we are removing them during our post-processing in the DICOM header data, therefore the algorithm is gaining no information from the header analysis to assist with the classification of text recognized in the pixel data as being a potential specific patient identifier. This is an issue that should be further investigated in a future iteration. One possible solution is to gather patient IDs during a pre-processing step and to use that information during de-identification.

Run using Source Dataset comparing MIDI Pipeline output with Source Dataset

Below are the discrepancies between the Source dataset and MIDI pipeline runs on the source dataset. There were too many image changes to perform a human review for each one, therefore, we performed a review on a random sample of 173 images. As a reminder, the Source Dataset contained roughly 18,000 more images than the TCIA dataset due to CTP having quarantined some due to the header information indicating a high risk of containing burned-in PHI/PII (e.g., being secondary capture images such as screenshots). This is where the discrepancy in the comparison between MIDI pipeline with the Source and UAMS curated datasets comes from.

As shown in Table 13, 173 images were inspected and 171 of them correctly had PHI/PII removed, resulting in a 1.2% false positive rate. Table 14 continues to explore the 173 instances based on what was correctly de-identified and what leaked. Of the 173 instances, there were 2 images with no PHI/PII (false positives) and 49 were completely successful in removing all PHI/PII. 120 images were partially de-identified, but still resulted in PHI/PII leaks. These included Patient IDs, Institution names and acronyms, and names. While a 28% success rate is well below satisfactory, 100% of the images examined were partially de-identified meaning a human-in-the-loop would have been flagged to inspect them based on that criterion. These are also all images that were quarantined by CTP, either as part of the data transfer process or as a pre-processing step.

Table 13. Sample of pixel data discrepancies comparing MIDI Pipeline and Source datasets

Sampled MIDI Pipeline vs Source False Positive Discrepancies	
Discrepancy Type	**Count - 173 instances**
PHI/PII Removal	171
MIDI Pipeline False Positive Image Removal	1
MIDI Pipeline Minor False Positive Image Removal	1

Table 14. Sample of pixel data PHI leaks comparing MIDI Pipeline and Source datasets

Sampled MIDI vs Source PHI/PII Leaks	
Type of PHI/PII Leak	**Count - 173 instances**
None	51/173 - 2 didn’t have PHIs, 49 completely successful
Patient ID	107
Place/Institution Acronym	89
Person Name	2

Partial Person Name De-identification	9

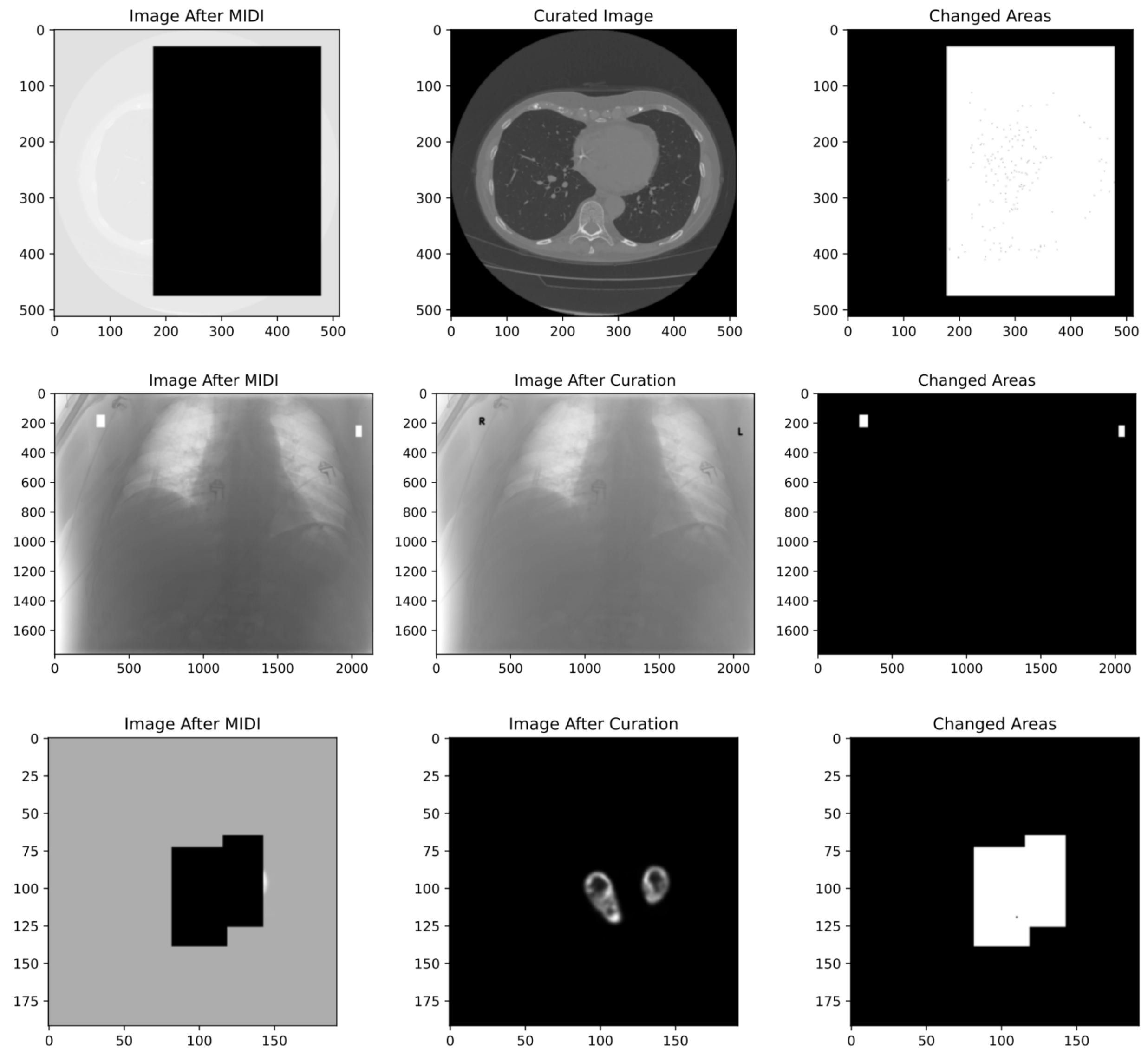


Figure 9. Examples of false positives found during MIDI pipeline runs.

Figure 9 are all examples of False Positives by the MIDI pipeline. In the first and third rows, the data has been majorly altered and would be unusable for analysis. In the middle row, the text ‘R’ and ‘L’ (important indicators of laterality) is incorrectly removed, but otherwise the pixel data remains intact.

IV.b Header Data

Run using MIDI-B Validation Dataset comparing MIDI Pipeline output with Answer Key

The following is a discussion of the validation script results of the final MIDI pipeline run on the MIDI-B Validation dataset. Table 15 details the unique header errors reported by the validation script, split into the six error types that the validation script reports. Notice a large number of errors are due to UIDs not being changed or not consistent. We discuss why this is the case in the below section titled “UID not Changed and UID not Consistent”. Since these errors would largely be fixed with a new run, we note both the total error percent with the UID errors (4.99%) and without the UID errors and actions (0.49%).

Table 15. Unique header error counts with MIDI-B Validation Dataset

Error Type	Count of Unique Errors	Total Required Actions	Percent Error (%)
Text not Removed	9	3036	0.29
Text not Retained	380	83368	0.45
Text not Null	50	5940	0.84
Tag not Retained	63	10455	0.60
UID not Changed	96	6402	1.50
UID not Consistent	5177	6402	80.87
Total	**5775**	**115603**	**4.99**
Total Without UID Errors or Actions	**502**	**102799**	**0.49**

Text not Removed

Table 16. Text not removed errors with MIDI-B Validation Dataset

Original File Value	De-Identified File Value	Data Element Name
LUNG CA ACCRIN 6668 at Harris Community Clinic	LUNG CA ACCRIN 6668 at Community Clinic	Additional Patient History
LUNG CA ACCRIN 6668 at	LUNG CA ACCRIN 6668 at	Additional Patient History

Harris Community Clinic	Community Clinic	
Admitted to Palmer-Greene Memorial on 20170803	Admitted to Memorial on 2017-06-29	Additional Patient History
PORTAL at Burke-Perkins Clinic	PORTAL at Clinic	Image Comments
Admitted to Alvarado, Atkins and Reid Memorial on 20170902	Admitted to and on 205-05-03	Image Comments
3MM VENOUS at Watkins, Brown and Reeves Community Clinic	3MM VENOUS at , and Community Clinic	Image Comments
Supine at Fuentes-Gibson Hospital	Supine at Hospital	Image Comments
MRI BREAT/UNILAT WO/WITH CON for Colleen Case	MRI BREAT/UNILAT WO/WITH CON for Case	Study Description
P4 for Sierra Townsend	P4 for Sierra	Study Description

Table 16 show the nine remaining text not removed errors are due to two error types; partially de-identified places (ex: 'Harris Community Clinic' being changed to 'Community Clinic') and partially de-identified names (ex: 'Sierra Townsend' being changed to 'Sierra'). The partially de-identified names are more serious errors than the partially de-identified places. These represent areas where the Google AI can still improve its de-identification algorithm.

Text Not Retained

Table 17. Text not retained errors by error type with MIDI-B Validation Dataset

Text not Retained Error Classification	Classification Subtype	Count	
Real Error	AI False Positive	91	124
	Post Processing False Positive	5	
	Incorrect Script Configuration	28	
Correct Action	DICOM Standard says to not retain the text	26	148

	Group decision to remove text to prevent PHI/PII leak	118	
	Text removed is actually PHI/PII	4	
Validation Script Error		108	
Total		380	

Real Errors
There are 124 errors of text not being retained that are considered real errors. These are primarily due to GCP incorrectly identifying something as a name or IP address and removing it. There are also five instances of the Post Processing script incorrectly identifying a string of integers as a patient ID within a data element. These types of errors are all represented by the below examples. There are also 28 unique instances of Referenced SOP Class UID being removed due to faulty configuration of the script. In future work, Referenced SOP Class UID should be moved to the Healthcare API script ‘keep’ list from the ‘remove’ list.

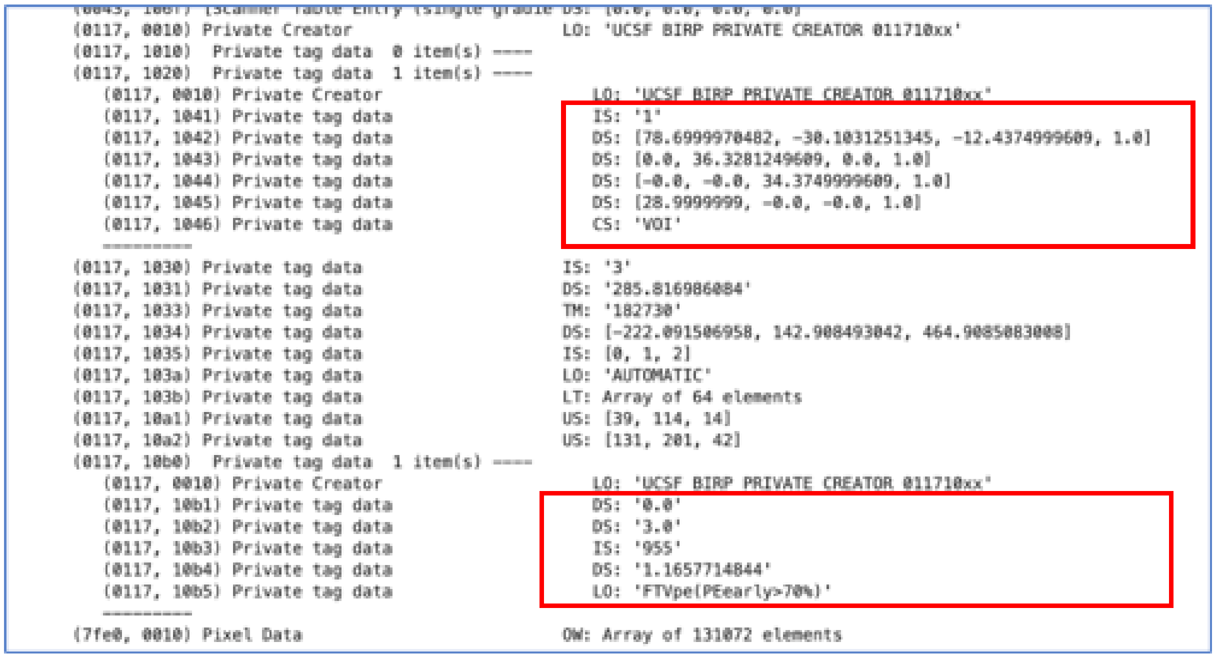
Table 18. Examples of real text not retained errors with MIDI-B Validation Dataset

Original File Value	De-Identified File Value	Tag Name
/BD/PRO Pelvis w&w/oCon at FMCC	/BD/PRO Pelvis w&w/1	Study Description
MR BREASTUNI UE for 744-55-9698	MR UE for	Study Description
2.3.0.81		Software Versions
[‘Rad:’, ‘hanning’, ‘7.00000 mm’]	[‘Rad:’, ‘’, ‘7.00000 mm’]	Convolution Kernel
Random Walker 3D	Random 3D	Code Meaning
8#200187150100100959010 0#30303040#1030	8#11#1#1030	Acquisition Device Processing Code
RM jamy brzusznej lub miednicy malej bez	RM brzusznej lub miednicy malej bez	Performed Procedure Step Description
CADstream 4.1.0.204	CADstream	Manufacturer’s Model Name

Validation Script Errors
The 108 Text not Retained errors come from private data elements that the validation script says are being incorrectly removed. We examined the de-identified files for all of these errors and found that these data elements are still present post de-identification, as shown in Figures 10 and 11 below. There is some sort of discrepancy in the validation script that is causing it to incorrectly interpret these elements as being removed when they aren't. The below figures show the errors reported, and the before and after files where the error supposedly takes place. The data elements are all still present in the de-identified file, as they should be.

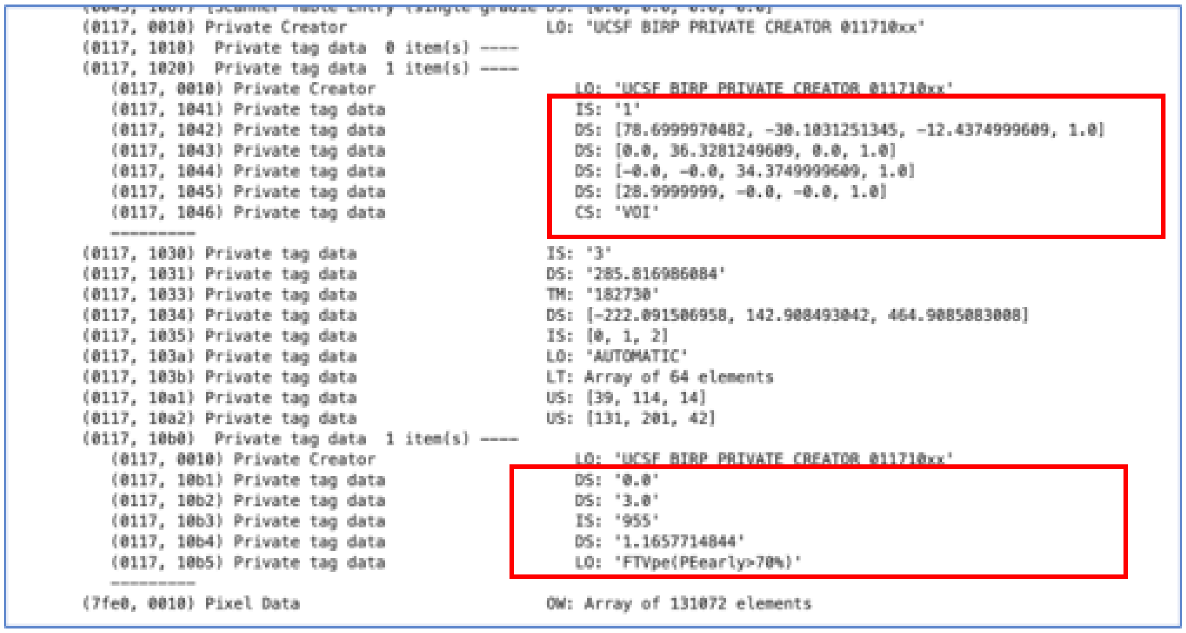
Table 19. Examples of validation script 'Text not Retained' errors with MIDI-B Validation Dataset

Original File Value	De-identified File Value	Tag Name
[]	1	Private tag data
[]	[91.100031618, -57.3031235132, 6.6406250391, 1.0]	Private tag data
[]	[0.0, 55.0781249609, 0.0, 1.0]	Private tag data
[]	VOI	Private tag data



```
(0043, 10a7) [Scanner Table Entry (single gradie DS: [0.0, 0.0, 0.0, 0.0]
(0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
(0117, 1010)  Private tag data  0 item(s) ----
(0117, 1020)  Private tag data  1 item(s) ----
   (0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
   (0117, 1041) Private tag data                     IS: '1'
   (0117, 1042) Private tag data                     DS: [78.6999970482, -30.1031251345, -12.4374999609, 1.0]
   (0117, 1043) Private tag data                     DS: [0.0, 36.3281249609, 0.0, 1.0]
   (0117, 1044) Private tag data                     DS: [-0.0, -0.0, 34.3749999609, 1.0]
   (0117, 1045) Private tag data                     DS: [28.9999999, -0.0, -0.0, 1.0]
   (0117, 1046) Private tag data                     CS: 'VOI'
   ---------
(0117, 1030) Private tag data                     IS: '3'
(0117, 1031) Private tag data                     DS: '285.816986084'
(0117, 1033) Private tag data                     TM: '182730'
(0117, 1034) Private tag data                     DS: [-222.091506958, 142.908493042, 464.9085083008]
(0117, 1035) Private tag data                     IS: [0, 1, 2]
(0117, 103a) Private tag data                     LO: 'AUTOMATIC'
(0117, 103b) Private tag data                     LT: Array of 64 elements
(0117, 10a1) Private tag data                     US: [39, 114, 14]
(0117, 10a2) Private tag data                     US: [131, 201, 42]
(0117, 10b0)  Private tag data  1 item(s) ----
   (0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
   (0117, 10b1) Private tag data                     DS: '0.0'
   (0117, 10b2) Private tag data                     DS: '3.0'
   (0117, 10b3) Private tag data                     IS: '955'
   (0117, 10b4) Private tag data                     DS: '1.1657714844'
   (0117, 10b5) Private tag data                     LO: 'FTVpe(PEearly>70%)'
   ---------
(7fe0, 0010) Pixel Data                           OW: Array of 131072 elements
```

Figure 10. File where Text not Retained errors supposedly occurred, before de-identification.



```
(0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
(0117, 1010)  Private tag data  0 item(s) ----
(0117, 1020)  Private tag data  1 item(s) ----
   (0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
   (0117, 1041) Private tag data                     IS: '1'
   (0117, 1042) Private tag data                     DS: [78.6999970482, -30.1031251345, -12.4374999609, 1.0]
   (0117, 1043) Private tag data                     DS: [0.0, 36.3281249609, 0.0, 1.0]
   (0117, 1044) Private tag data                     DS: [-0.0, -0.0, 34.3749999609, 1.0]
   (0117, 1045) Private tag data                     DS: [28.9999999, -0.0, -0.0, 1.0]
   (0117, 1046) Private tag data                     CS: 'VOI'
   ---------
(0117, 1030) Private tag data                     IS: '3'
(0117, 1031) Private tag data                     DS: '285.816986084'
(0117, 1033) Private tag data                     TM: '182730'
(0117, 1034) Private tag data                     DS: [-222.091506958, 142.908493042, 464.9085083008]
(0117, 1035) Private tag data                     IS: [0, 1, 2]
(0117, 103a) Private tag data                     LO: 'AUTOMATIC'
(0117, 103b) Private tag data                     LT: Array of 64 elements
(0117, 10a1) Private tag data                     US: [39, 114, 14]
(0117, 10a2) Private tag data                     US: [131, 201, 42]
(0117, 10b0)  Private tag data  1 item(s) ----
   (0117, 0010) Private Creator                      LO: 'UCSF BIRP PRIVATE CREATOR 011710xx'
   (0117, 10b1) Private tag data                     DS: '0.0'
   (0117, 10b2) Private tag data                     DS: '3.0'
   (0117, 10b3) Private tag data                     IS: '955'
   (0117, 10b4) Private tag data                     DS: '1.1657714844'
   (0117, 10b5) Private tag data                     LO: 'FTVpe(PEearly>70%)'
   ---------
(7fe0, 0010) Pixel Data                           OW: Array of 131072 elements
```

Figure 11. File where Text not Retained errors supposedly occurred, after de-identification.

Correct Actions

Table 20. Text not retained actions incorrectly labeled as failures by validation script with MIDI-B Validation Dataset

Correct Action Subtype	Action Count
DICOM Standard says to remove	26
Group decision made to remove	118
PHI/PII Removal	4
Total	148

The 148 Text not Retained 'Correct Actions' can be further broken down into three subclasses as seen in Table 20. The first are data elements being removed that, according to the DICOM Standard, should be removed. These elements are all (with the exception of Gantry ID and Detector ID) in the Icon Image Sequence. It is unclear why the TCIA validation script is flagging errors when these elements are removed when, according to the DICOM Standard, that is the correct way to handle them. There are 26 unique 'errors' of this type.

The second class of ‘Correct Actions’ refers to instances where the MIDI pipeline removes values in data elements that don’t need to be removed and are usually kept, but were added to the remove list for the MIDI pipeline based on group consensus. All 118 of these ‘errors’ come from the Text Value data element. In Phase 3 of the MIDI project, we saw multiple instances of person initials appearing in the synthetic MIDI-B datasets (it should be noted that the MIDI datasets were created based on TCIA’s experience with where they have encountered PHI/PII in DICOM files in the past). These initials are present without context (e.g., ‘RS’), making it very challenging to identify them as initials. It was determined by the group that since Text Value almost never contains information crucial to scientific research, the best way to deal with this issue is to remove all text in Text Value. This results in 118 ‘errors’.

The third and final class of ‘Correct Actions’ is instances where the MIDI pipeline removes PHI/PII and the validation script incorrectly identifies this as a *Text not Retained* error. There are only four examples of this. One example is in Protocol Name, where “4.6 COLONOSCOPY (ACRIN) DR.IYER for Nicholas Gomez” is changed to “4.6 COLONOSCOPY (ACRIN) for “ by the MIDI pipeline. Here the MIDI pipeline is removing a patient name and a doctor name. However, the validation script identifies this as a *Text not Retained* error.

Tag not Retained and Text not Null

There are 63 “Tag not Retained" errors. These are data elements that according to the DICOM standard should always be present in a DICOM file. These errors all come from data elements that weren’t present in the original input files. Since our de-identification process does not create these data elements, errors are flagged by the validation process. However, this is an issue with the original data, not a result of the de-identification process.
50 errors are “Text not null” errors. These are for data elements that according to the DICOM standard should not be empty. These errors are all caused due to the same reason as the tag not retained errors, because of the data elements being empty or absent in the original files, not because of the de-identification process. We confirmed this by running the validation script on the original, non-de-identified dataset which resulted in exactly the same number of “Tag not Retained” and “Text not Null” errors.

UID not Changed and UID not Consistent

A large majority of the UID errors were due to the issue with files being dropped by the Healthcare API discussed in the ‘Results: Dropped Files’ section. This issue was fixed by Google during the later part of Phase 4, so our last run of the MIDI-B Validation dataset was done before this fix. As a result, there are many reported errors for the MIDI-B Validation dataset that wouldn’t be present if the pipeline were run again today. This is evidenced by the MIDI-B Test dataset run discussed in the section below, which originally saw the same kinds of UID errors due to the dropped files issue and no longer had those errors when we reran the pipeline on that dataset after Google fixed the issue.

The UID errors are all due to issues with how the mapping files were created, not issues with the de-identification capability of the pipeline. All 96 UID not Changed errors were due to dropped files. 540 files were being dropped during the GCP de-identification stage of the pipeline due to an issue with the Operation Metadata feature discussed in the Dropped Files section below. Our workaround for this issue was to run a version of the pipeline without Operation Metadata on the 540 files being dropped. In order to still include these files in the UID mapping CSV file, we had to change this version of the pipeline to not change the three main mapping UIDs (Study, Series, and SOP Instance UID). We then manually added these unchanged UIDs to the UID mapping file. Because these UIDs were unchanged for the dropped files, we got 96 UID not Changed errors on these three UIDs. The dropped files workaround also resulted in 5177 "UID not Consistent" errors. "UID not Consistent" errors were calculated by looking at the UID mapping file and checking that the UIDs before and after match the UIDs in the DICOM files. In the main run of the pipeline, the Operation Metadata feature tracks changes made to UIDs and then creates the UID mapping file accordingly. However, in the secondary run on the dropped files, changes made to UIDs are not tracked in the UID mapping file due to the lack of the Operation Metadata feature. This results in 4850 "UID not Consistent" errors. The remaining 327 "UID not Consistent" errors are from UIDs in private data elements. We handle private data elements in the post processing script by automatically keeping all private data elements listed as safe by TCIA and automatically removing the rest (more details can be found in section Methods: Post Processing Scripts). This results in 327 private UIDs being removed. The post processing script isn't configured to track these changes in the mapping files, so this results in UID not Consistent errors.

Run using MIDI-B Test Dataset comparing MIDI Pipeline output with answer key

The following results are from the most recent run of the MIDI pipeline on the MIDI-B Test Dataset.

Text not removed

There are 33 unique *Text not Removed* errors. All except one of these are partially de-identified hospital names (ex:Mack Memorial changed to just 'Memorial'). The only other *Text not Removed* error is a Referenced File ID that isn't being removed. This can be fixed in the future by adding Referenced File ID to the Healthcare API script 'remove' list.

Text not retained

The validation script flagged 674 of the unique changes made by the MIDI pipeline to the DICOM files as *Text not Retained* errors. However, closer examination by the team determined that not all these changes were actually errors. The MIDI team went through all of the errors and classified them as either 'real errors', 'correct actions', 'validation script errors' or 'unknown'. Of the 674 *Text not* Retained errors, 195 were classified as 'real errors', 270 were classified as ‘correct actions’, 151 were classified as ‘validation script errors’, and 58 were classified as ‘unknown’.

Table 21. Text not retained unique changes flagged by validation script broken down into real errors, correct actions, validation script errors, and unknown.

Text not Retained Error Classification	Classification Subtype	Count	
Real Error	AI False Positive	142	195
	Post Processing False Positive	14	
	Incorrect Script Configuration	39	
Correct Action	DICOM Standard says to not retain the text	70	328
	Group decision to remove text to prevent PHI/PII leak	250	
	Text removed is actually PHI/PII	8	
Validation Script Error		151	
Total Text not Retained Errors		674	

Correct Actions

The 270 'Correct Actions' can be further categorized into three subclasses. The first involves data elements that should be removed according to the DICOM Standard. All of these elements, except for Gantry ID and Detector ID, are located in the Icon Image Sequence. It is unclear why the TCIA validation script is flagging errors when these elements are removed, as this action aligns with the DICOM Standard. There are 70 distinct 'errors' of this type.

The second category of ‘Correct Actions’ pertains to cases where the MIDI pipeline removes values from data elements that don't actually need to be removed and are typically retained. These elements were added to the removal list for the MIDI pipeline based on group consensus. All 192 of these ‘errors’ stem from the Text Value data element. As discussed in the Comparison of MIDI pipeline output with Answer Key using MIDI-B Validation Dataset section, the group decided to remove all instances of Text Value.

The third and final category of ‘Correct Actions’ involves instances where the MIDI pipeline removes PHI/PII, but the validation script mistakenly flags this as a Text not Retained error. There are only eight such cases. One example occurs in Reconstruction Method, where "3D Kinahan - Rogers" is changed to "3D" by the MIDI pipeline. The pipeline is correctly removing a name, but the validation script incorrectly classifies this as a Text not Retained error.

Real Errors:

The ‘Real Errors’ can be split into three subclasses. The first subclass is data elements removed due to the script configuration being incorrect. All 39 of these errors are from Referenced SOP Class UID being changed. Unlike other UIDs, class UIDs should not be changed at all during de-identification. All other class UIDs were put in the ‘keep list’ of the GCP script, so they were not changed. Referenced SOP Class UID was not. This configuration should be changed in future versions of the script, which would remove these errors.

The second subclass of ‘Real Errors’ is of errors due to the GCP AI incorrectly identifying and then removing PHI/PII that is not actually PHI/PII. As stated previously in this report, the pipeline is split into the GCP Healthcare API phase and the post processing phase. The GCP Healthcare API handles, among other things, searching through free text values to find person names and IP addresses. The 142 GCP AI false positives all come from Google’s AI identifying and removing names or IP addresses that are not actually names or IP addresses. For instance, in Software Versions, the value ‘3.3.1.35’ is removed since the Google AI identifies it incorrectly as an IP address. In Study Description, ‘MR Breast’ is changed to ‘MR ‘ since the Google AI reads ‘MR’ as ‘Mr’ and then identifies ‘Breast’ as a last name.

The third and final subclass of ‘Real Errors’ is of errors due to the post processing script identifying and then removing PHI/PII that is not actually PHI/PII. The post processing script, among other things, handles removing Patient IDs and shifting dates. There are 14 post processing false positive errors. They are due to two types of errors. The first is the post processing script incorrectly identifying something as a patient ID and then removing it. The script is configured to identify any strings of integers longer than 6 but less than 15 (with the exception of integers strings within specific data elements where we would expect to find long integers strings that are not patient IDs) and replace the strings with ‘1’ (this is done so that we can easily see where the post processing script found a patient ID for testing purposes. Future versions of the pipeline can change this to replacing the patient ID with a blank space instead). The group consensus is that strings of integers over 6 but less than 15 (with the exception of the tags previously discussed) are almost always going to be patient IDs, and won't contain any crucial research information. However, there are still a small number of resulting false positives. For instance, in Series Description 'Nodule 6 - Annotation 114086 evaluations' is changed to 'Nodule 6 - Annotation 1 evaluations'. The second type of post processing false positive error is from the post processing script incorrectly identifying something as a date and then shifting it. For instance in Image Comments, '<(5033/11/185)-(5033/11/9)>' is changed to '<(5033/11/185)-(501932-02-20)>' due to the post processing script identifying '33/11/9' as November 9th, 1933. This could be fixed in future script versions by changing the date finder function to not identify something as a date if it is a subsection of a string of integers ('5033/11/9' should not be identified as a date due to the 50 leading the 33).

Validation Errors:

The 151 errors come from Private tag data elements that the validation script says are being incorrectly removed. We examined the de-identified files for all of these errors and found that these tags are still present post de-identification. There is some sort of error in the validation script that is causing it to incorrectly interpret these elements as being removed when they aren't. The below figures show the errors reported, and the before and after files where the error supposedly takes place. The data elements are all still present in the de-identified file.

Tag not Retained and Text not Null

There are 89 tag not retained errors. These are tags that according to the DICOM standard should always be present in a DICOM file. These errors all come from tags that weren't present in the original input files. Since our de-identification process does not create these tags, errors are flagged by the validation process. However, this is an issue with the original data, not a result of the de-identification process.

71 errors are text not null errors. These are for tags that according to the DICOM standard should not be empty. These errors are all caused due to the same reason as the tag not retained errors, because of the tags being empty or absent in the original files, not because of the de-identification process. We confirmed this by running the validation script on the original, non-de-identified dataset which resulted in the same exact number of tag not retained and text not null errors.

UID not Changed and UID not Consistent

The UID errors are all due to issues with how the mapping files were created, not issues with the de-identification capability of the pipeline. The only UID not changed error occurs in the private data element PET pulse_frame_id. This UID is most likely not changed due to the VR not being a UID VR.

The remaining 365 UID not Consistent errors are from private UIDs. We handle private data elements in the post processing script by automatically keeping all private data elements listed as safe by TCIA and automatically removing the rest (more details can be found in section Methods: Post Processing Scripts). This results in 365 private UIDs being removed. The post processing script isn't configured to track these changes in the mapping files, so this results in UID not Consistent errors.

Run using Source Dataset comparing MIDI Pipeline output with UAMS Curated Dataset

The following are the Comparison Framework results of the final MIDI pipeline run on the source dataset. The Comparison Framework collects all the discrepancies between the Source dataset and MIDI pipeline outputs. Due to the massive size of the dataset, we aggregated the results into 'Unique Discrepancies'. For example, in a Study Description, the original value '*MR_COMPLETE$123456$*' is changed by UAMS to '*MR_COMPLETE*' and by MIDI pipeline to '*MR_COMPLETE$1$*' (since 1 is the replacement value for patient IDs in the MIDI pipeline) 555 times. This corresponds to 555 total discrepancies and one unique discrepancy. Table 22 identifies the number of unique discrepancies.

Table 22. Unique count of discrepancies between UAMS Curated dataset and MIDI pipeline output using the Source Dataset

Discrepancy Type	Unique Discrepancy Count
Correct Action	6102
False Negative (but not PHI leak)	847
False Positive	918
PHI Leak	21
Grand Total	**7888**

False Negatives

There were 868 unique false negatives. Of these 868 false negatives, 21 of them were potential PHI leaks. The potential PHI leaks fall into the following two categories:

- In Date of Calibration, there was a list of dates which were left unshifted due to the post-processing script not being formulated to look through lists within data elements. This can be fixed going forward by updating the post-processing script.

- In Image Type, there was one instance of a datetime that wasn't caught by the post-processing because the Image Type is a Code String VR. The post-processing script does not look through Code String VRs, as Code String VRs were not identified as a VR where PHI/PII might be found. This should be re-evaluated going forward.
- In Protocol Name, 'UAMS' was left in the element value. The issue of hospital/medical center acronyms not being removed was one we also saw in the synthetic MIDI dataset results.

The remaining 847 false negatives do not contain any potential PHI leaks, but rather are data elements that should be removed for other reasons, such as to maintain formatting or dataset integrity issues. 22 of these are Overlay and Curve data elements according to the DICOM Standard, these should always be removed, unless the Clean Graphics Option is specified and supported. The MIDI pipeline removes the values, but does not remove the data elements entirely, which is not in accordance with DICOM Standard. This issue, along with all other 842 false negatives not containing PHI, can be fixed in future iterations of this work by adding these data elements to the Healthcare API script 'remove' list.

False Positives

There are 918 unique false positive discrepancies. They are all detailed below in Table 23 below. The most common examples of false positives are from the post-processing script falsely identifying patient IDs or dates, or the GCP de-identification API falsely identifying names. For instance, in Manufacturer Model's Name, 'Aurora' is removed because the GCP API identifies it as a name. In Instance Number, the post processing script removed '1320000' because it identifies it as a patient ID (Instance Number is very rarely so many digits, so it is unusual for this kind of false positive to occur). Another false positive we have seen before in the synthetic MIDI datasets is the GCP API falsely identifying IP Addresses. In Software Version, '5.3.1\5.3.1.3' is changed to '5.3.1\' due to the GCP API identifying '5.3.1.3' as an IP address. Of the 918 False Positive discrepancies, 713 are due to issues with the Healthcare API's AI and 266 are due to issues in the Post Processing script.

Table 23. False positives being reported by tag name

Tag Name	Error Count	Tag DS	Source Value	MIDI Value	TCIA Value	Reason for False Positive
Patient Orientation Code Sequence-Context Group Version	42950	(0054,0410)-(0008,0106)	20020904000000.000000	20010524000	20020904000000.000000	This is a date being shifted, however according the the DICOM Standard Context Group Version should not be altered

Radiopharmaceutical Information Sequence-Context Group Version	28016	(0054,0016)-(0008,0106)	20020904000000.000000	20000901000	20020904000000.000000	This is a date being shifted, however according the the DICOM Standard Context Group Version should not be altered
Patient Gantry Relationship Code Sequence-Context Group Version	21475	(0054,0414)-(0008,0106)	20020904000000.000000	20010920000	20020904000000.000000	This is a date being shifted, however according the the DICOM Standard Context Group Version should not be altered
Software Version(s)	17398	(0018,1020)	5.3.1\5.3.1.3	['5.3.1', '']	5.3.1\5.3.1.3	GCP API incorrectly identifies an IP address
Manufacturer's Model Name	1253	(0008,1090)	Aurora II		Aurora II	GCP API incorrectly identifies a person name
Content Sequence-Text Value	210	(0040,A730)-(0040,A160)	Mammo 8.3	1	Mammo 8.3	The group made the decision to replace Text Value with 1 to avoid potential PHI leaks
Secondary Capture Device Software Version(s)	149	(0018,1019)	8.4.3.16		8.4.3.16	GCP API incorrectly identifies an IP address
Study Description	9	(0008,1030)	IR CVL > 5 YRS OLD	IR CVL >	IR CVL > 5 YRS OLD	GCP API is configured to remove the 'age' infotype. This can be changed in the future
Procedure Code Sequence-Code Meaning	9	(0008,1032)-(0008,0104)	IR CVL > 5 YRS OLD	IR CVL >	IR CVL > 5 YRS OLD	GCP API is configured to remove the 'age' infotype. This can be changed in the future
Instance Number	4	(0020,0013)	1320000	1	1320000	Post processing script incorrectly identifies a patient id
Detector Information Sequence-Collimator/grid Name	4	(0054,0022)-(0018,1180)	LEHR		LEHR	GCP API incorrectly identifies a person name

Acquisition Device Processing Code	2	(0018,1401)	8#1351151051001009590100#404000#1040#20350#35220	8#11#1#1040#20350#35220	8#1351151051001009590100#404000#1040#20350#35220	post processing incorrectly identifies a patient id
Detector Description	2	(0018,7006)	8-03.00	1998-03-30	8-03.00	Post processing script incorrectly identifies and then shifts a date
Series Description	2237	(0008,103E)	;;;Coronal.38/Coronal	;;;Coronal./Coronal	;;;.38/Coronal	Healthcare API falsely identifies a name

Correct Actions

Table 24. Unique correct actions taken by MIDI pipeline based on subtype that were flagged as discrepancies between MIDI pipeline output and UAMS Curated Dataset.

Correct Action Subtype	Unique Correct Action Count
MIDI Pipeline Correctly Removing	16
MIDI Pipeline Correctly Retaining	969
UAMS Curation	1223
Expected Discrepancy	3848
Formatting	46
Total	**6102**

There are 6102 unique correct actions as shown in Table 24. These can be further split into five subcategories, which are discussed below:

- **MIDI PipelineCorrectly Removing** discrepancies refer to instances where MIDI pipeline correctly removes something that UAMS misses.
- **MIDI Pipeline Correctly Retaining** discrepancies refer to instances where MIDI pipeline safely retains something that UAMS removes. Some of these instances are because MIDI pipeline's AI algorithm is often more specific with PHI/PII removal than UAMS. For instance, UAMS commonly removes entirely free text values that contain PHI rather than selectively editing the text, while the MIDI pipeline de-identifies the values and retains more information. As an example, Study Description may contain: *'[3/10/2008 11:29 AM DOE JOHN]CT LUMBAR SPINE   DR. SMITH 123-456-7899   HISTORY OF MULTIPLE MYELOMA   AND L4-L5 INFECTION. PATIENT STATES INFECTION GONE LAST JUNE12   PLEASE EVALUATE L 4-5 FOR FUSION . HERNIATED DISK'*

(**Note**: This has been de-identified from the original). While UAMS removes this text in its entirety, MIDI pipeline de-identifies the text, changing it to: *'[2007-04-23 11:29 AM]CT LUMBAR SPINE HISTORY OF MULTIPLE MYELOMA   AND L4-L5 INFECTION. PATIENT STATES INFECTION GONE LAST 0000-07-16   PLEASE EVALUATE L 4-5 FOR FUSION . HERNIATED DISK'.*

- **UAMS Curation** discrepancies are due to UAMS changing values for curation purposes. For instance, there are many instances of UAMS changing Body Part Examined due to the value not accurately reflecting the body part in the corresponding image. There are also many times where UAMS changes a value for DICOM conformance reasons. For example, in the element Laterality, UAMS removes the value 'B' as it is not a valid value for the Laterality element. The MIDI pipeline was built purely to de-identify data, not curate or correct DICOM conformance issues in data, so it leaves these values alone. For the purposes of this project (which is to evaluate the de-identification ability of the MIDI pipeline), this action is correct. In future phases, the MIDI pipeline could be built out further to do many of the aspects of curation and DICOM conformance that UAMS does
- **Excepted Discrepancies** are discrepancies where UAMS curation and MIDI pipeline differ in ways expected. The main example of this is in date shifting. Both UAMS curation and MIDI pipeline shift dates, but using different seeds and ranges, so the end result is different. Despite the end results being different, both UAMS curation and MIDI pipeline's actions are correct. The header metadata also contains values where de-identification should produce different results. For instance, Source Application Entity Title should be changed to specify the tool used for de-identification. Since UAMS curation and MIDI pipeline use different de-identification tools, these values are different. There are also 16 unique instances where the value of Text Value is changed to '1'. As discussed in the MIDI-B Dataset Results sections, this was a decision made by the group in order to prevent PHI/PII leaks.
- **Formatting** discrepancies are examples where the UAMS curation and MIDI pipeline values are essentially the same save for minor formatting changes. For example, both UAMS curation and MIDI pipeline remove a name from the string in a Protocol Name data element. However, MIDI pipeline leaves an additional white space in place of the name, resulting in a minor difference between the two values.

Run using CTP Dataset comparing MIDI Pipeline output with UAMS Curated Dataset

The following are the Comparison Framework results of the final MIDI pipeline run on the CTP dataset seen in Table 25. The below tables show the unique and total discrepancy count between the UAMS Curated dataset and the MIDI pipeline output. Since the dataset had already been run through the CTP, certain de-identification aspects of the MIDI pipeline were turned off for this run. For instance, dates in 'DA' and 'DT' data elements did not require shifting as they had been shifted by CTP.

Table 25. Comparison Framework results of the final MIDI pipeline run on the CTP dataset.

Discrepancy Type	Unique Discrepancy Count
False Positive	697
False Negative (but not PHI/PII leak)	834
PHI/PII Leak	13
Correct Action	7170
Unknown	787
Total	**9501**

Unknown

There are 787 unique unknown discrepancies. These are most likely due to an issue with duplicate UIDs in the dataset. The original Source dataset contained around 300 files with duplicate UIDs. This led to the Comparison Framework to sometimes compare files that are not actually the same files, resulting in reported discrepancies that do not accurately reflect the contents of the MIDI pipeline and UAMS curated outputs. The issue with duplicate files was compensated for in the future versions of the Comparison Framework that were run on the Source dataset, so these issues were not present there.

False Negatives

The False Negatives are mostly the same as from the Source Results. The slightly smaller number of discrepancies is most likely due to the CTP dataset being a smaller subset of the Source Dataset.

PHI/PII Leak

The PHI/PII Leaks in the CTP dataset are all due to the error where ‘UAMS’ is left in Protocol Name. This error is discussed more in the Source Results section. The CTP results do not contain the Date of Calibration errors due to the relevant files not being in the CTP dataset.

False Positives

Table 26. False positives by subcategory of either Healthcare API or post processing errors.

False Positive Subcategories	Unique False Positive Count
Healthcare API false positive	690

Post Processing false positive	7
Total	**697**

There were 697 False Positives which are all a subset of the False Positives discussed in the Source Dataset Results section. 690 of these are due to issues with the Healthcare API's AI (such as the AI incorrectly identifying Software Versions as IP addresses) and 7 are issues from the Post Processing script, such as incorrectly removing long strings of integers due to mistaking them for patient identifiers. These issues are all discussed further in the Source Dataset Results section.The decrease in False Positives from the Source Dataset to the CTP Dataset is likely due to the CTP dataset being a smaller subset of the Source Dataset.

Correct Actions

Table 27. Correct actions taken that were flagged by comparison framework

Correct Action Subtype	Unique Correct Action Count
MIDI Pipeline Correct Removal	5
MIDI Pipeline Correct Retaining	335
UAMS Curation	1223
Expected Discrepancy	5556
Formatting	45
PHI/PII removed	6
Total	**7170**

The 'Correct Actions' are mostly the same as discussed in the Source Dataset Results section. There are more Expected Discrepancies than in the Source Dataset Comparison due to private data elements being examined in the Comparison Framework when the CTP Dataset was examined, but not when the Source Dataset was examined. As discussed in the Methods: Post Processing Scripts section, MIDI pipeline removes all private data elements that are not known to be safe to keep. UAMS curation still keeps some of these elements, and this flags 5534 unique instances of private data elements being removed by the MIDI pipeline but kept by UAMS curation. Other than this, there is an overall decrease in 'Correct Action' discrepancies. This could be due to the CTP dataset being smaller than the Source dataset, meaning all else equal, there would be fewer discrepancies in the CTP comparison than the Source comparison. The other main change from the Source results is that there are six instances of PHI/PII being left by UAMS curation that the MIDI pipeline catches and removes. For example, in an instance of Performed Procedure Step Description contains a date that the MIDI pipeline finds and removes, but UAMS curation leaves this value untouched.

IV.c Dropped Files

During pipeline runs of the Source dataset, files were being dropped during the Healthcare API run. It was a total of 540 images, a small subset of the whole dataset. It was identified to be linked to how we track the UID changes through the use of the FHIR store and a FHIR output. When this option was turned off the images ran fine. However, this resulted in the 540 images not having their UIDs changed, increasing our error rate. This was reported to Google and identified as a code error on their backend. They were able to complete the fix and push the code for us to rerun the pipeline. At time of this publication, this fix is part of the public beta version that we describe here.

After this fix by Google, there remained a smaller subset of 26 images still failing to pass through the pipeline at the point of the Healthcare API. This has been identified as a Transcoding Error in the logs. This has been reported to Google, but it is unclear what the issue is at the time of writing. We are attempting to provide the team with example images that are clean of PHI/PII so they can replicate the issue.

One final dropped files issue encountered was a large number of files being dropped without logging during one of our runs of the pipeline. This was fixed by rerunning the dropped files. We are assuming that this is due to the large number of files we attempted to run all at once, as smaller subsets of data never encountered this issue. This is something to be aware of in any operational deployment; any runs of the pipeline should check that the total number of files in the output equals the number in the input.

IV.d Run Performance

For the Synthetic datasets, runtimes of the Healthcare API took around 5 minutes to complete, and post-processing script runtimes took around 10 minutes. For the Source dataset, which contains 393,899 files, the Healthcare API runtime took about 8 minutes and the post-processing script took about an entire week to complete. This is partially due to the manual intervention needed to run the post-processing script. The post-processing script would time out after an hour and required the user to manually hit go to continue running it. We parallelized this process, allowing the user to run separate batches of files all at once. With this fix, the post-processing script was able to run on the Source dataset in an hour.

IV.e Cost Analysis

The cost of the MIDI pipeline is dependent on a few factors. This includes the total number of files being de-identified, number of elements inspected, number of elements transformed, and how much and how long data is stored. Google provides a detailed guide on pricing[26].

As an example of the cost we incurred, for the Source dataset containing 393,899 instances, it cost $572.62 for Healthcare API use, $41.81 for compute (analysis), and $5.36 for Cloud Run Functions (post-processing) as shown in Figure 12. This results in a total cost of $619.79 per run or $1.57 per 1000 images.

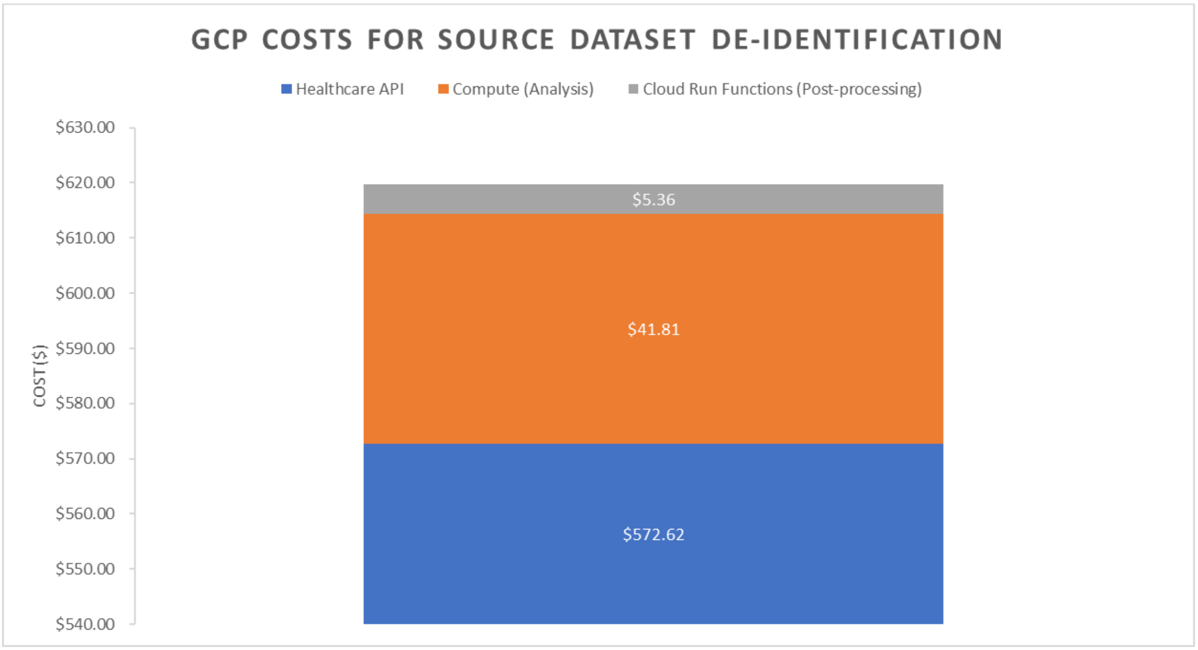


Figure 12. Cost for a single MIDI pipeline run of the Source Dataset.

V. Future Considerations & Recommendations

De-identification of medical imaging data, a legal requirement, is vitally important for data sharing, educational training, and the advancement of research, for example, when training machine learning and artificial intelligence algorithms which require large volumes of training data. Through this project and the MIDI-B challenge we observed different processes attempting to accurately remove PHI/PII from images. However, in all cases, no method had a 100% success rate and some small amounts of PHI/PII were still able to leak through. This means that in a production setting, a human-in-the-loop is required to review the cases of false positives and fields of concern for leaked PHI/PII (i.e., known issues of false negatives), in order to achieve the desired threshold of accuracy of 100%, especially when it comes to removing PHI/PII. We believe that the MIDI pipeline described here achieves a level of accuracy that makes it possible to configure a production setting with an appropriate human-in-the-loop process that substantially reduces the human workload.

We also believe that the MIDI pipeline is sufficiently accurate for the purpose of checking whether submitted data (e.g., to NCI's IDC) has been properly de-identified. The Healthcare API logs all changes in an image using the FHIR store which can then be analyzed in BigQuery. The post-processing script also makes changes to images and can be logged as csv files, which will need to be fully configured and tested in future runs. Otherwise, all changes made to images can be analyzed as has been done in this stage by doing a direct comparison between input images and the output de-identified images. Since most pipeline inaccuracies are fairly predictable and limited to specific data elements, these can be easily monitored. Therefore, a run of the MIDI pipeline, configured with a human-in-the-loop process, should be able to reliably capture any residual PHI/PII. Images will be flagged based on criteria such as pixels removed in the middle of images or names being identified and removed. The human-in-the-loop would be able to take these flagged images and do a manual review to identify any concerns with the submitted images. If deemed by the human-in-the-loop that the submitted images are too risky then they can be rejected and sent back to the submitter for further de-identification. This will substantially reduce the risk of PHI/PII leakage on datasets published in environments such as IDC. It will also greatly reduce the time it would take for a human to review a full dataset of images, reducing costs due to this decrease in manual processing.

For the underlying Google Healthcare API, there remain a few areas of concern that should be addressed. First, there are a large number of fixes that have been implemented through the development of post-processing scripts. See above section on De-Identification Scripts for more details. These are all items that could be integrated into future releases of the Healthcare API itself and Google has added these items to their roadmap. As the Healthcare API releases updated versions with new possible configurations then the post-processing scripts can be phased out. This will improve the efficiency of the platform as the post-processing currently takes the longest time to complete.

Second, since the Google Healthcare API is part of a service that is an ensemble of aggregated microservices, functions are not static and will see changes that should lead to improvements in future releases. However, after each new release of the Healthcare API, the MIDI pipeline should be regression tested before deploying the new release. The addition of version controls to Google's Healthcare API would be highly beneficial to allow control over deployment of releases, to make switching to new releases an intentional part of the regular maintenance process.

Third, the configurations that we use are still considered part of the Healthcare API's beta release, even if they are publicly available. Eventually, they should be made part of a Generally Available (GA) release of the Healthcare API.

All requests made to the Google team for improvements to the Healthcare API are provided below in section VI. Additional Information.

A potential future production release of the MIDI pipeline at NCI, and indeed at NIH, could include the following services: (1) a verification service for curators (human-in-the-loop) of data repositories or data commons (e.g., IDC) to check for residual PHI/PII in submitted datasets before publication; (2) a service for end users to check for PHI/PII in their medical imaging datasets or before publishing to the public. Both services should include an appropriate interactive user interface that allows these users to be the human-in-the-loop.

Our most recent work also tested to a small extent Generative AI (GenAI) tools to identify PHI/PII in strings. Where the pipeline currently falls short, such as with patient identifiers and various acronyms, a GenAI tool could be utilized. Since the MIDI pipeline is built completely within the Google Cloud using Cloud Functions for execution, a GenAI based step using Google's VertexAI environment could be configured and included in the MIDI pipeline to review data element text values that are considered high risk for PHI/PII. This could be incorporated currently as part of post-processing or as a future release by Google making GenAI an option in their Healthcare API.

As for any deployments of products or services in a government IT environment, the MIDI platform will first need to obtain an Authority to Operate (ATO). This will require working closely with the NCI CBIIT Security Team (or the NIH equivalent).

Finally, it would also be of use to develop an easy-to-use front-end so users do not need to be knowledgeable of the Google Cloud Console to run the MIDI pipeline. We envision a tool that allows users to connect to their GCP environment and invoke the necessary commands with a click of a button. In this tool, the user will be able to select from options that specify the actions they want to perform, for which tags, and for which data stored in Cloud Buckets. In addition, there will be sets of preconfigured configurations that match with the Confidentiality Profiles defined in DICOM Standard PS3.15[5]. This service will help investigators quickly de-identify their data without needing to go through the hassle of accessing the GCP console or needing to do any terminal commands, making this tool much more accessible and reliable. In addition, this same interface could be used to analyze results of the de-identification process, making it a one-stop solution for end-to-end de-identification. This service could be provided as part of the account setup or as a stand alone CBIIT application, giving NCI users the ability to submit for an approved de-identification run on a shared MIDI pipeline account.

In summary, the above suggestions outline specific steps that would now allow to take the MIDI pipeline to a production setting, thus making it usable for the desired use cases, specifically, as a tool to validate submissions to IDC, as long as the submitted data have been deidentified based on a consistent public protocol such as TCIAs, which our pipeline is tested against.

VI. Additional Information

Special Thanks To Everyone Who Supported MIDI Phase 4

Granger Sutton, Keyvan Farahani, Nick Weber, David Clunie, Ulrike Wagner, Qinyan Pan, Scott Gustafson, Michael Rutherford, Linmin Pei, Tracy Nolan, Kirk Smith, Tanja Davidsen, Erica Kim, Dale Hawkins, Cheryl Corman, Patrice Walters, Dave Belardo, Antej Nuhanović, Juergen Klenk,

Ben Kopchick, Theresa Do, Kathryn Johnson, and Laura Opsahl-Ong. If anyone was missed, we apologize, but thank you for your support.

GitHub Links

Here we provide all code repository links found in GitHub for ease of reference.

1. Cloud Functions scripts (includes Healthcare API configuration and post processing code)
2. Post processing notebooks
3. Validation Scripts

MIDI Task Group

Mission and Goals:

- To document strategies and best practices in medical image de-ID for secondary sharing of imaging data with an emphasis on DICOM
- To reach consensus on best practices
- To disseminate findings
- To provide input toward CBIIT/NCI and other ICs activities
- To make recommendations on criteria and resources for performance evaluation of tools
- To provide guidelines for image de-ID using automated vs. manual, cloud-based vs. local approaches, portability, scalability

See also the Report of the Medical Image De-Identification (MIDI) Task Group -- Best Practices and Recommendations[24].

Requests Made to Google For Improving Healthcare API

Recommended Healthcare API Improvements	Description of Improvement
Improve Date Shifting	Dates in yyyymmdd format that aren't in DA type elements aren't identified as dates. This should be changed. Also, do not allow for a zero shift. And potentially allow the user to input desired range of shift.
Improve Patient ID detection	Patient IDs are currently often not being identified and removed in both header and pixel data. The post-processing script finds Patient IDs in header data by searching for strings of integers of a length greater than 6. We recommend the Healthcare API implement this, or some other more

	advanced form of patient ID identification.
Improve Acronym Detection (potentially with use of a white-list/black-list)	Hospital acronyms and some person initials are currently not being identified and removed by the Healthcare API. The post processing script mostly fixes this by using regular expressions (ex: 'at XXX' or 'by XX'). However, allowing the user to input a custom white or black list could also help improve this issue.
Update Healthcare API to correctly handle private data elements.	Private Creator elements, which if changed in any way result in the loss of all other private elements, are sometimes being altered when the algorithm identifies possible PHI in them. It is currently challenging to specify that Private Creator elements be kept, since there is a large range of possible hexadecimal tags that could represent Private Creator, and Google does not have a way of specifying private elements by name. Also, private data elements require the corresponding Private Creator to be correctly identified. However, the Healthcare API only allows users to be specified by tag name or tag hexadecimal.
Improve general person name detection. Uncommon names are often missed, or sometimes only partially de-identified (EX: Sierra Case is changed to just Case)	Improve general person name detection. Uncommon names are often missed, or sometimes only partially de-identified (EX: Sierra Case is changed to just Case)
Versioning control	It would be very difficult to use this in production without the guarantee that the algorithm's behavior isn't changing without our ability to test those changes. The user should at least be notified when major changes occur so they can retest the pipeline.
Allow user to change AI temperature	Allow the user to change the AI temperature, either on the entire de-identification tool or specific Info Types.

	For instance, if the dataset being de-identified has many uncommon patient names, the user could turn the temperature up on the Person Name infotype so that the Healthcare API would widen its net for Person Name de-identification.
Implement a flag for human review feature	In cases where PHI is hard to automatically remove without also removing lots of non-PHI information (non-traditional date formats, patient IDs, acronyms etc), the Healthcare API could be configured to flag these kinds of potential cases for human review. Potentially allow the user to specify things at a specific confidence interval be flagged (ex: if the AI is 70% sure that a string is a person name, this should be flagged).
Group 0x0002 file meta data should be removed and regenerated in a specific way, not just de-identified	The way it is done now is contrary to the Dicom Standard. For instance, there are certain group 0x0002 elements that need to be updated to reflect the de-identification tool and practices used.

References

1. Monteiro E, Costa C, Oliveira JL. A De-Identification Pipeline for Ultrasound Medical Images in DICOM Format. J Med Syst. 2017 May;41(5):89. doi:10.1007/s10916-017-0736-1.

2. 1. Khin K, Burckhardt P, Padman R. A Deep Learning Architecture for De-identification of Patient Notes: Implementation and Evaluation. arXiv; 2018. Available from: http://arxiv.org/abs/1810.01570.
3. https://www.cancerimagingarchive.net/
4. Pianykh, O. S. (2012). Digital imaging and communications in medicine (DICOM): a practical introduction and survival guide, 2nd Edition. Springer. ISBN 978-3-642-10849-5
5. National Electrical Manufacturers Association (NEMA). Digital Imaging and Communications in Medicine (DICOM) Standard PS3.15 - Security and System Management Profiles - E.1 Attribute Confidentiality Profiles - De-identifier. Rosslyn, VA: National Electrical Manufacturers Association (NEMA); Available from: http://dicom.nema.org/medical/dicom/current/output/chtml/part15/chapter_E.html#sect_E.1.1
6. Moore SM, Maffitt DR, Smith KE, Kirby JS, Clark KW, Freymann JB, et al. De-identification of Medical Images with Retention of Scientific Research Value. RadioGraphics. 2015 May 1;35(3):727–35. doi:10.1148/rg.2015140244.
7. Douglass MM, Cliffford GD, Reisner A, Long WJ, Moody GB, Mark RG. De-identification algorithm for free-text nursing notes. In: Computers in Cardiology, 2005. 2005. p. 331–4. doi:10.1109/CIC.2005.1588104.
8. Douglass M, Clifford GD, Reisner A, Moody GB, Mark RG. Computer-assisted de-identification of free text in the MIMIC II database. In: Computers in Cardiology, 2004. 2004. p. 341–4. Available from: http://ecg.mit.edu/george/publications/douglass-cinc-2004.pdf doi:10.1109/CIC.2004.1442942.

9. Hartman T, Howell MD, Dean J, Hoory S, Slyper R, Laish I, et al. Customization scenarios for de-identification of clinical notes. BMC Med Inform Decis Mak. 2020 Jan 30;20(1):14. doi:10.1186/s12911-020-1026-2.
10. https://cloud.google.com/healthcare/docs/how-tos/dicom-deidentify
11. https://cloud.google.com/dlp
12. Kopchick B, Klenk J, Carlson T, Kumpatla M, Klimov S, Mikdadi D, et al. Medical image de-identification using cloud services. In: Park BJ, Deserno TM, editors. Medical Imaging 2022: Imaging Informatics for Healthcare, Research, and Applications. San Diego, United States: SPIE; 2022. p. 19. Available from: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12037/2608972/Medical-image-de-identification-using-cloud-services/10.1117/12.2608972.full doi:10.1117/12.2608972
13. Kopchick BP, Opsahl-Ong LK, Pan Q, Rutherford MW, Wagner U, Singh BS, et al. Accelerating de-identification of images with cloud services to support data sharing in cancer research. In: AACR Annual Meeting. 2023. Available from: http://wiki.nci.nih.gov/display/MIDI/AACR+Poster
14. National Electrical Manufacturers Association (NEMA). DICOM Standard. DICOM. Available from: http://www.dicomstandard.org/
15. https://cloud.google.com/healthcare-api/docs/concepts/dicom
16. https://cloud.google.com/healthcare-api/docs/how-tos/dicom-best-practices
17. Rutherford M, Mun SK, Levine B, Bennett W, Smith K, Farmer P, et al. A DICOM dataset for evaluation of medical image de-identification. Sci Data. 2021 Jul 16;8(1):183. doi:10.1038/s41597-021-00967-y

18. Rutherford M, Mun SK, Levine B, Bennett WC, Smith K, Farmer P, et al. A DICOM dataset for evaluation of medical image de-identification (Pseudo-PHI-DICOM-Data). The Cancer Imaging Archive; 2020. Available from: https://wiki.cancerimagingarchive.net/x/MYDTB doi:10.7937/S17Z-R072
19. http://synapse.org/Synapse:syn53065760/wiki/
20. Freymann JB, Kirby JS, Perry JH, Clunie DA, Jaffe CC. Image Data Sharing for Biomedical Research—Meeting HIPAA Requirements for De-identification. J Digit Imaging. 2012 Feb;25(1):14–24. doi:10.1007/s10278-011-9422-x
21. https://cloud.google.com/consulting/portfolio/isolator-secure-sensitive-data-for-collaboration
22. https://pydicom.github.io/
23. https://github.com/CBIIT/MIDI_validation_script
24. Clunie DA, Flanders A, Taylor A, Erickson B, Bialecki B, Brundage D, et al. Report of the Medical Image De-Identification (MIDI) Task Group -- Best Practices and Recommendations [Internet]. arXiv; 2023. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10081345/ doi:10.48550/arXiv.2303.10473
25. https://www.synapse.org/Synapse:syn53065760/wiki/625274
26. https://cloud.google.com/healthcare-api/pricing

Appendix

Medical Image De-Identification (MIDI) Standard Operation Procedure

Update: 11/27/2024

This article serves as a Standard Operating Procedure (SOP) for running the Medical Image De-Identification (MIDI) Pipeline hosted in the CBIIT managed Google Cloud Platform (GCP) environment. The audience of this SOP are members with access to the MIDI pipeline and approved to run the pipeline.

Requirement/Prerequisites

- The current GCP account the pipeline is setup in is *nih-nci-cbiit-midi-dev2*
  - To access the MIDI pipeline hosted on CBIIT's GCP platform, users must have access to GCP and the nih_nci_cbiit_imaging_informatics_midi_dev_folder_admin@nih.gov distribution list.
  - For access, a request can be made through a Google Cloud Service Request.
- User will require the following IAM permissions either through the user list or a service account:
  - Storage Admin
  - Cloud Functions Invoker
  - Cloud Run Invoker
  - Vertex AI Service Agent
- The required APIs that need activated:
  - Cloud Run Functions
  - Healthcare API
  - Bigquery

Getting Started – Account Setup

If you are an NCI employee looking to deploy this within the NCI Cloud Environment, please submit a consultation request using the Google Cloud Service Request. The Cloud Engineering team will assist you in setting up the environment with the listed above permissions and required APIs.

If you are using a personal account, please proceed to Cloud Computing Services | Google Cloud to setup a free account. If you are new to GCP, we highly recommend you take some tutorials to familiarize yourself with GCP console and terminology. In your personal account you will have administrator privileges and will be able to proceed to activating the necessary APIs. Otherwise, you will need the proper IAM permissions listed in the Requirements section. To activate APIs, click the hamburger button in the top left and select “View All Products”. Then find “Cloud Run functions” in the Serverless section and select it. It is also recommended to pin the product for easy access later. Then select the “Enable” button to enable the product. Proceed to do this with the “Healthcare API” and “Bigquery”.

The final step is to copy the pipeline functions in to Cloud Run.

1. In “Cloud Run functions” select “Create Function”. Give function a name such as *De-identification*.
2. Select the Region you would like it to run in. For NCI, it should use us-east4 (Northern Virginia).
3. Trigger will be HTTPS, though other triggers can be explored.
4. In “Runtime”, select the necessary settings for Memory and CPU. We recommend with 1GB memory and 1 CPU to start with. This can be edited in the future as needed.
5. Set “Timeout” to the maximum setting of 3600 seconds.
    a. This determines how long a job can run in Cloud Functions. Since the process can take a long time, we maximize this. In addition, this determines how you will need to split up your jobs depending on number of files you are de-identifying.
6. Concurrency should be 1
    a. This determines how many requests can run on the same instance. To run in parallel, we want this to be 1 so that new instances are spun up instead of attempting to run on the same instance.
7. “Autoscaling” we recommend to have a minimum of 0 and a maximum of 100.
    a. This allows 100 instances to run at the same time for this function.
8. “Runtime Service Account” should be the accounts service account that has the minimum permissions to run the pipelines and the user should also have permission to run as the service account.
    a. More details about service accounts can be found here: https://cloud.google.com/iam/docs/service-account-overview.

Click next to go to Code. Select a version of Python, anything Python 3.8 or greater will work. Copy over *main.py, requirements.txt,* and *script.py*. In *main.py* make the necessary edits to “project” and “location”. Other variables are described below. The *requirements.txt* and *script.py* do not need to be edited. Please proceed below to run the pipeline.

Procedure

The pipeline is run through two GCP Cloud Run Functions. See below for more information.

1. *De-identification*
2. *Post-processing*

The functions take the process through linear steps of:

1. Storage
2. Heatlhcare DICOM Dataset/Datastore
3. Healthcare De-Identification
4. De-identified Healthcare DICOM Dataset/Datastore and FHIR store
5. De-identified Storage
6. Post-processing

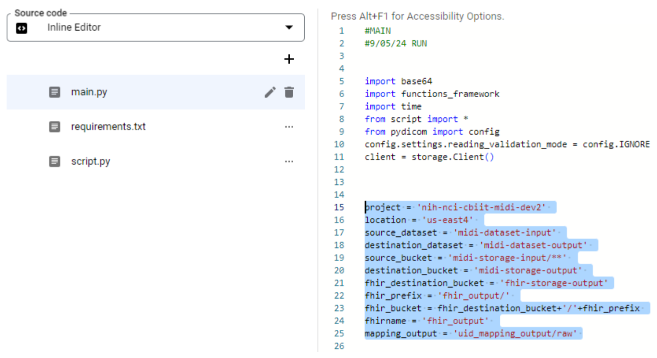
Steps

Bucket Creation

1. Go to **Cloud Storage** in the GCP console
2. DICOM data needs to be uploaded into a bucket. At NCI, created buckets should be in a *single region* in *us-east4*.
    a. Data can be uploaded by dragging and dropping or through an API command.
    b. For purposes of this SOP, we will use the bucket *midi-storage-input*
3. DICOM output bucket needs to be created.
    a. For purposes of this SOP, we will use the bucket *midi-storage-output*
4. FHIR output bucket needs to be created.
    a. For purposes of this SOP, we will use the bucket *fhir_output*

Cloud Run Function 1: De-Identification using Healthcare API

1. Go to **Cloud Run functions** in the GCP console
2. Select *de-identification-v1*
3. Go to the **Source** tab
4. In the main.py file check that all variables are correct



```
#MAIN
#9/05/24 RUN


import base64
import functions_framework
import time
from script import *
from pydicom import config
config.settings.reading_validation_mode = config.IGNORE
client = storage.Client()



project = 'nih-nci-cbiit-midi-dev2'
location = 'us-east4'
source_dataset = 'midi-dataset-input'
destination_dataset = 'midi-dataset-output'
source_bucket = 'midi-storage-input/**'
destination_bucket = 'midi-storage-output'
fhir_destination_bucket = 'fhir-storage-output'
fhir_prefix = 'fhir_output/'
fhir_bucket = fhir_destination_bucket+'/'+fhir_prefix
fhirname = 'fhir_output'
mapping_output = 'uid_mapping_output/raw'

```

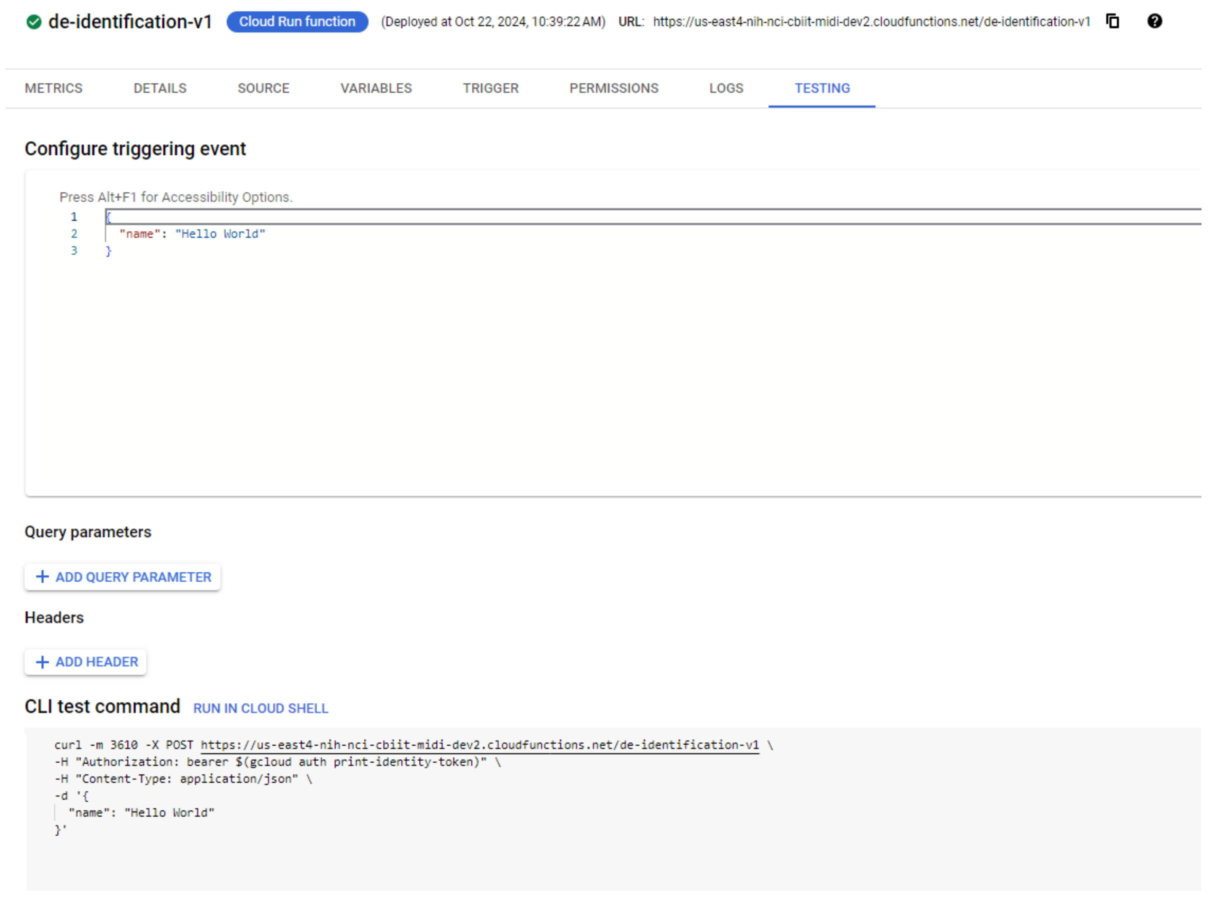
    a. project: Name of the GCP project
    b. location: Location of your buckets and Healthcare stores
        i. location should be 'us-east4'
    c. source_dataset: Name of Healthcare datastore you want created to upload raw data
        i. Cannot be already existing

- *d.* destination_dataset: Name of Healthcare datastore de-identified data will be exported to
    - *i.* *Cannot already be existing*
- *e.* source_bucket: Bucket created in Bucket Creation 2 where raw data was uploaded
    - *i.* The '/**' at end of bucket takes all files in bucket for de-identification
    - *ii.* You can add a path to file or path to folder in bucket to not take all files
        - *1.* Ex. source_bucket = 'midi-storage-input/patient1/**'
- *f.* destination_bucket: Bucket created in Bucket Creation 3
- *g.* fhir_destination_bucket: Bucket you want fhir output to be exported to
- *h.* fhir_prefix: Folder the fhir output will be placed in in the fhir_destination_bucket
- *i.* fhir_bucket: Ignore
- *j.* fhirname: Healthcare FHIR store created when script is run
    - *i.* *Cannot already be existing*
- *k.* mapping_output: Folder inside fhir_destination_bucket where UID mapping file will reside

5. Variables can be edited by selecting **EDIT**
6. After variables are correct, select **SAVE AND REDEPLOY**
    - *a.* This will take a couple of minutes
7. To run the pipeline open the Cloud Shell or a terminal in an AI Notebook
8. Run the following curl command:

```
curl -m 3610 -X POST https://us-east4-nih-nci-cbiit-midi-dev2.cloudfunctions.net/de-
identification-v1 \
-H "Authorization: bearer $(gcloud auth print-identity-token)" \
-H "Content-Type: application/json" \
-d '{
  "name": "Hello World"
}'
```

9. This can be found in the **Testing** tab under CLI test command.



de-identification-v1 Cloud Run function (Deployed at Oct 22, 2024, 10:39:22 AM) URL: https://us-east4-nih-nci-cbiit-midi-dev2.cloudfunctions.net/de-identification-v1

Configure triggering event

Press Alt+F1 for Accessibility Options.

```
{
  "name": "Hello World"
}
```

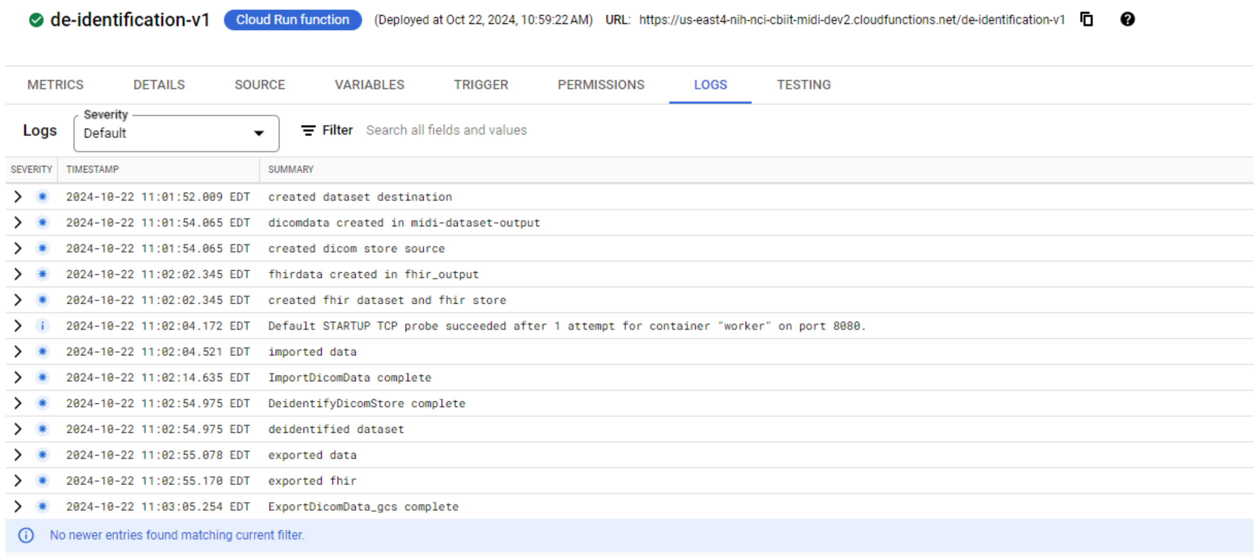
Query parameters

Headers

CLI test command

```
curl -m 3610 -X POST https://us-east4-nih-nci-cbiit-midi-dev2.cloudfunctions.net/de-identification-v1 \
-H "Authorization: bearer $(gcloud auth print-identity-token)" \
-H "Content-Type: application/json" \
-d '{
  "name": "Hello World"
}'
```

10. You can follow the progress of the script under the **Logs** tab



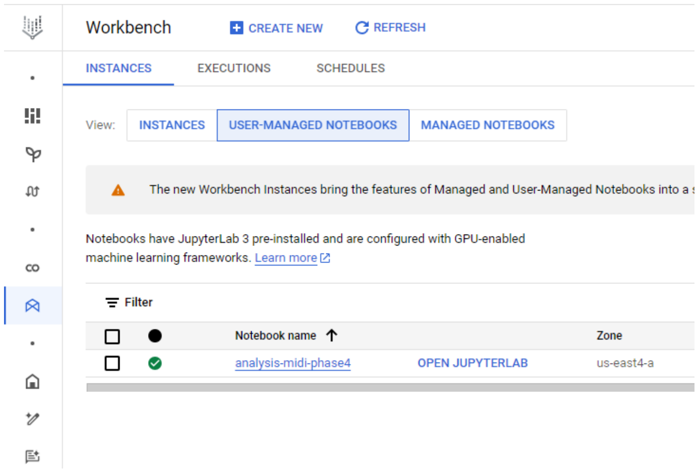
de-identification-v1 Cloud Run function (Deployed at Oct 22, 2024, 10:59:22 AM) URL: https://us-east4-nih-nci-cbiit-midi-dev2.cloudfunctions.net/de-identification-v1

Logs Severity: Default Filter Search all fields and values

```
SEVERITY TIMESTAMP                    SUMMARY
         2024-10-22 11:01:52.009 EDT  created dataset destination
         2024-10-22 11:01:54.065 EDT  dicomdata created in midi-dataset-output
         2024-10-22 11:01:54.065 EDT  created dicom store source
         2024-10-22 11:02:02.345 EDT  fhirdata created in fhir_output
         2024-10-22 11:02:02.345 EDT  created fhir dataset and fhir store
    i    2024-10-22 11:02:04.172 EDT  Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.
         2024-10-22 11:02:04.521 EDT  imported data
         2024-10-22 11:02:14.635 EDT  ImportDicomData complete
         2024-10-22 11:02:54.975 EDT  DeidentifyDicomStore complete
         2024-10-22 11:02:54.975 EDT  deidentified dataset
         2024-10-22 11:02:55.078 EDT  exported data
         2024-10-22 11:02:55.170 EDT  exported fhir
         2024-10-22 11:03:05.254 EDT  ExportDicomData_gcs complete
```

No newer entries found matching current filter.

11. After successful completion, check the destination buckets for the final outputs
12. Files can be checked by navigating to a Jupyter Notebook environment.
    a. **VertexAI** > **Workbench** > **User-Managed Notebooks**



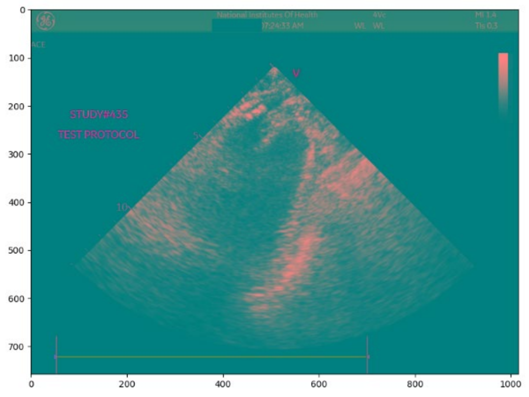
The new Workbench Instances bring the features of Managed and User-Managed Notebooks into a

Notebooks have JupyterLab 3 pre-installed and are configured with GPU-enabled machine learning frameworks. Learn more

		Notebook name ↑		Zone
☐	✓	analysis-midi-phase4	OPEN JUPYTERLAB	us-east4-a

b. In the *analysis-midi-phase4* notebook, we have a demo.ipynb notebook that showcases a quick way to look at some of the images using pydicom

```
[18]: dicom_out.pixel_array.shape

[18]: (92, 758, 1016, 3)

[19]: plt.figure(figsize = (10,10))
      plt.imshow(dicom_out.pixel_array[0])

[19]: <matplotlib.image.AxesImage at 0x7f87e445df30>
```



Cloud Run Function 2: Post Processing Script

1. Go to **Cloud Run functions** in the GCP console
2. Select *post-processing-v1*
3. Go to the **Source** tab
4. In the main.py file check that all variables are correct

```python
def hello_http(request):
    request_json = request.get_json()
    filestart = int(request_json["filestart"])
    fileend = int(request_json["fileend"])
    #check for post processing progress
    bucketname = 'midi-storage-output'
    bucket = client.get_bucket(bucketname)
    folder_in = 'healthcareapi-out/'
    filelist_in = [item.name for item in bucket.list_blobs(prefix=folder_in)]
    #filelist_in = filelist_in[1:]
    #filelist_out = [item.name.split('/')[-1] for item in bucket_out.list_blobs(prefix='func_postp/output2')]
    folder_out = 'postprocessing-out/'
    filelist_out = [item.name.split(folder_out)[-1] for item in bucket.list_blobs(prefix=folder_out)]
```

   a. bucketname: Bucket where your results are from *de-identification-v1 run*
   b. folder_in: Folder in bucket where files are located from *de-identification-v1 run*
   c. folder_out: Name of folder you want postprocessing results to end up in
5. Variables can be edited by selecting **EDIT**
6. After variables are correct, select **SAVE AND REDEPLOY**
   a. This will take a couple of minutes
7. To run the pipeline open the Cloud Shell or a terminal in an AI Notebook
8. Run the following curl command:

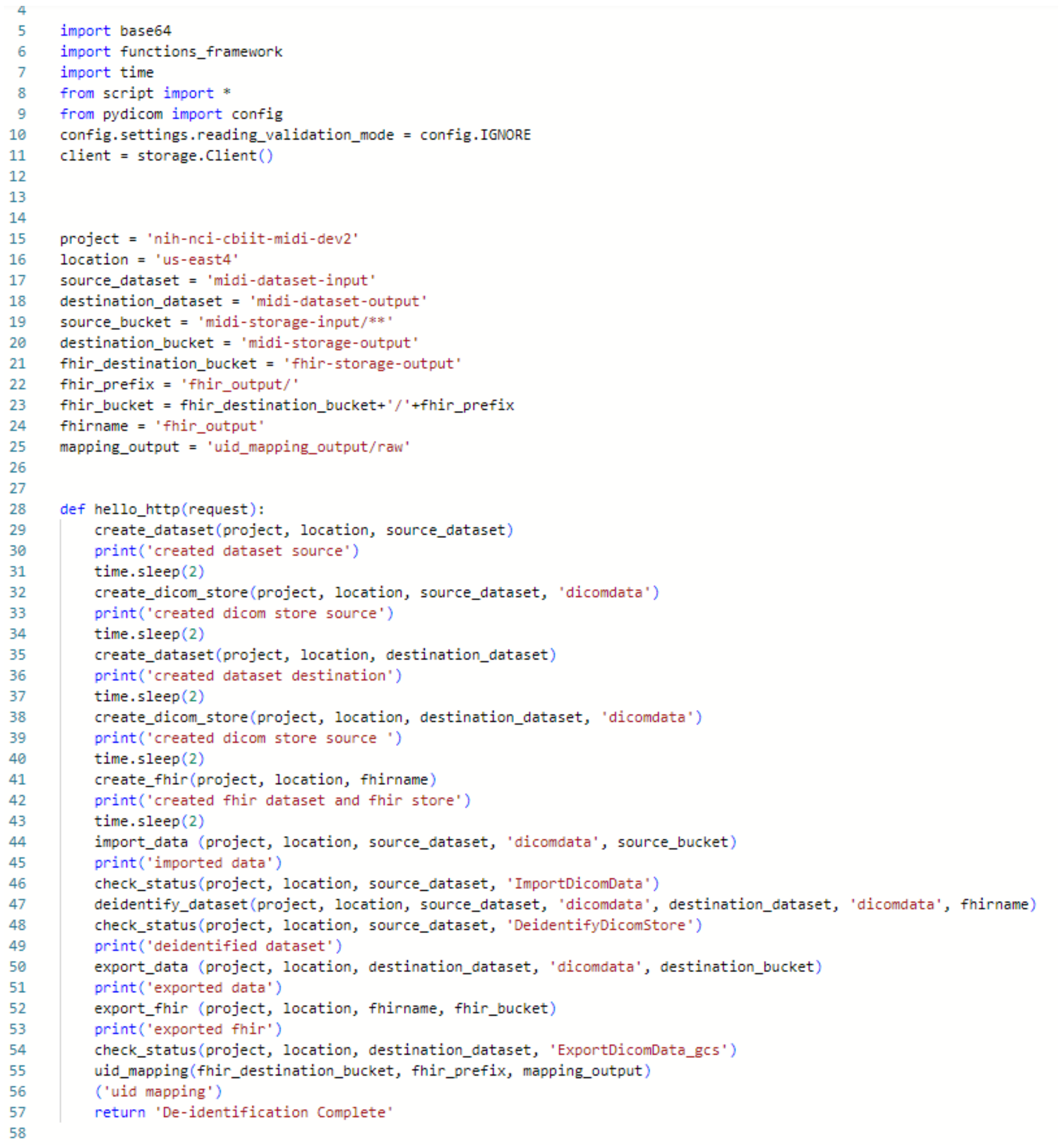
```
curl -m 3610 -X POST https://us-east4-nih-nci-cbiit-midi-dev2.cloudfunctions.net/post-processing-v1 -H "Authorization: bearer $(gcloud auth print-identity-token)" -H "Content-Type: application/json" -d '{

  "name": "Hello World", "filestart": "0", "fileend": "1"

}'
```

9. Edit the filestart and fileend variables to match with which files you want to use. This can be used to parallelize the runs. For example: If you have 500 files, you can run in one terminal *"filestart": "0", "fileend": "249"* and in another terminal *"filestart": "250", "fileend": "500"* to complete the task twice as fast.
   a. filestart: 0 – Integer that determines which file to start with
   b. fileend: 1 – Integer that determines which file to end with
10. You can follow the progress of the script under the **Logs** tab
11. Check files in bucketname/folder_out to make sure all files are present.

Code Deep Dive

The code is split into 3 python scipts: *main.py*, *script.py*, and *requirements.txt*.

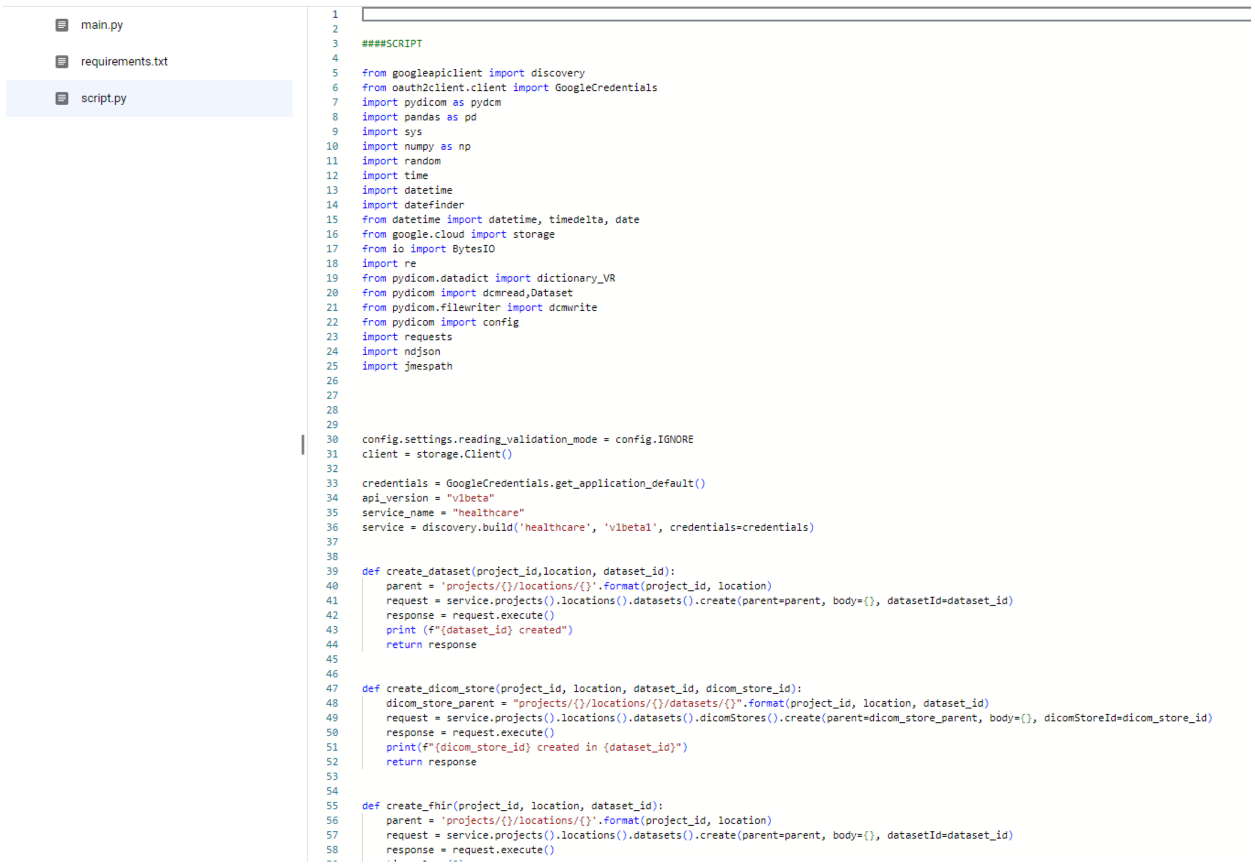
main.py



```
import base64
import functions_framework
import time
from script import *
from pydicom import config
config.settings.reading_validation_mode = config.IGNORE
client = storage.Client()



project = 'nih-nci-cbiit-midi-dev2'
location = 'us-east4'
source_dataset = 'midi-dataset-input'
destination_dataset = 'midi-dataset-output'
source_bucket = 'midi-storage-input/**'
destination_bucket = 'midi-storage-output'
fhir_destination_bucket = 'fhir-storage-output'
fhir_prefix = 'fhir_output/'
fhir_bucket = fhir_destination_bucket+'/'+fhir_prefix
fhirname = 'fhir_output'
mapping_output = 'uid_mapping_output/raw'


def hello_http(request):
    create_dataset(project, location, source_dataset)
    print('created dataset source')
    time.sleep(2)
    create_dicom_store(project, location, source_dataset, 'dicomdata')
    print('created dicom store source')
    time.sleep(2)
    create_dataset(project, location, destination_dataset)
    print('created dataset destination')
    time.sleep(2)
    create_dicom_store(project, location, destination_dataset, 'dicomdata')
    print('created dicom store source ')
    time.sleep(2)
    create_fhir(project, location, fhirname)
    print('created fhir dataset and fhir store')
    time.sleep(2)
    import_data (project, location, source_dataset, 'dicomdata', source_bucket)
    print('imported data')
    check_status(project, location, source_dataset, 'ImportDicomData')
    deidentify_dataset(project, location, source_dataset, 'dicomdata', destination_dataset, 'dicomdata', fhirname)
    check_status(project, location, source_dataset, 'DeidentifyDicomStore')
    print('deidentified dataset')
    export_data (project, location, destination_dataset, 'dicomdata', destination_bucket)
    print('exported data')
    export_fhir (project, location, fhirname, fhir_bucket)
    print('exported fhir')
    check_status(project, location, destination_dataset, 'ExportDicomData_gcs')
    uid_mapping(fhir_destination_bucket, fhir_prefix, mapping_output)
    ('uid mapping')
    return 'De-identification Complete'

```

- This code is where all the variables for bucket names and datasets are defined and are found near the top after the package imports.
- The code then follows through a step-by-step process to create the necessary datasets and dicom/fhir stores. It then imports data from the defined source bucket. The deidentify_dataset() function then runs the de-identification script that calls the Healthcare API function.
- Data is then exported to the de-identification bucket and fhir output bucket.
- A UID mapping is created and put in the fhir output bucket

script.py



```python


####SCRIPT

from googleapiclient import discovery
from oauth2client.client import GoogleCredentials
import pydicom as pydcm
import pandas as pd
import sys
import numpy as np
import random
import time
import datetime
import datefinder
from datetime import datetime, timedelta, date
from google.cloud import storage
from io import BytesIO
import re
from pydicom.datadict import dictionary_VR
from pydicom import dcmread,Dataset
from pydicom.filewriter import dcmwrite
from pydicom import config
import requests
import ndjson
import jmespath




config.settings.reading_validation_mode = config.IGNORE
client = storage.Client()

credentials = GoogleCredentials.get_application_default()
api_version = "v1beta"
service_name = "healthcare"
service = discovery.build('healthcare', 'v1beta1', credentials=credentials)


def create_dataset(project_id,location, dataset_id):
    parent = 'projects/{}/locations/{}'.format(project_id, location)
    request = service.projects().locations().datasets().create(parent=parent, body={}, datasetId=dataset_id)
    response = request.execute()
    print (f"{dataset_id} created")
    return response


def create_dicom_store(project_id, location, dataset_id, dicom_store_id):
    dicom_store_parent = "projects/{}/locations/{}/datasets/{}".format(project_id, location, dataset_id)
    request = service.projects().locations().datasets().dicomStores().create(parent=dicom_store_parent, body={}, dicomStoreId=dicom_store_id)
    response = request.execute()
    print(f"{dicom_store_id} created in {dataset_id}")
    return response


def create_fhir(project_id, location, dataset_id):
    parent = 'projects/{}/locations/{}'.format(project_id, location)
    request = service.projects().locations().datasets().create(parent=parent, body={}, datasetId=dataset_id)
    response = request.execute()
```

script.py contains all the functions that are called by main.py.

The main part of the script is the deidentify_dataset function. This contains the Healthcare API call that is configured to this project. It is important to note the structure of this request under the dicomTagConfig options:

```
'dicomTagConfig' : {
        "actions" : [
                {"queries": [
                        'List of DICOM tags', 'Ex. 00091008', 'DT'
                        ],
                "action": {}
                }
        ]
}
```

The red is what is customized. Actions can include: keepTag, removeTag, resetTag, cleanTextTag, cleanImageTag, recurseTag, and regenUidTag. In addition, we specify a profile type of *DEIDENTIFY_TAG_CONTENTS*' and that we are collecting the metadata of changed files through the fhir store by activating the *operationMetadata* option. More details about all these actions can be found here: https://cloud.google.com/healthcare-api/docs/how-tos/dicom-deidentify-dicomtagconfig