8-2021

Conversion of MultiCellDS Digital Snapshots to ISA-Tab format.

Corey P. Chitwood
University of Louisville

Recommended Citation

Chitwood, Corey P., "Conversion of MultiCellDS Digital Snapshots to ISA-Tab format." (2021). *Electronic Theses and Dissertations.* Paper 3914.
https://doi.org/10.18297/etd/3914

CONVERSION OF MULTICELLDS DIGITAL SNAPSHOTS TO ISA-TAB FORMAT

By
Corey Chitwood
B.S., University of Louisville, May 2020

A Thesis
Submitted to the Faculty of the
University of Louisville
J.B. Speed School of Engineering
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Bioengineering

July 2021

CONVERSION OF MULTICELLDS DIGITAL SNAPSHOTS TO ISA-TAB STANDARD

Submitted by: ___________________________________

Corey Chitwood

A Thesis Approved On

_________________July 19, 2021__________________

By the Following Reading and Examination Committee:

____

Dr. Hermann Frieboes, Thesis Director

____

Dr. Joseph Chen

____

Dr. Nihat Altiparmak

ACKNOWLEDGEMENTS

This Thesis Project would not have been possible without several people who helped me along the way, both during and prior to this project.

To Dr. Hermann Frieboes, for asking me to take on this project. While I had not originally planned on completing a Thesis project, I am glad that I did, and feel that I have gained a tremendous amount of experience and gained new skills. I also appreciate you taking time out of your schedule to meet with Reid, Connor, and I every other week for the past year. During the most chaotic of years, those meetings helped keep me on track with my project and keep a positive outlook thanks to a laugh or two.

To Dr. Sam Friedman and Dr. Paul Aiyetan, for always being there to answer my questions and keep me on the right track to completing the project.

To Dr. Joseph Chen and Dr. Nihat Altiparmak for serving on my Thesis Examination Committee.

And to Sarah, for being there to support me and keep me on track

ABSTRACT

The MultiCellular Data Standard (MultiCellDS) is an interdisciplinary effort to create a data standard for sharing multicellular experimental, simulation, and clinical data. The ultimate goal of the overall project is to allow for data sharing that will lead to better analyses and simulations for multicellular biology and predictive medicine in association with the PhysiCell, a software that allows for simulations of large numbers of cells in 3D tissues. Digital cell lines are files that contain a standardized representation of a biological cell line and include phenotypic parameters as well as microenvironmental conditions for use in simulations. A Digital Cell Line is a data model rather than a computational model, meaning that it is based on curated measurements of specific cells in certain conditions. A Digital Snapshot is a recording of the current state of an experiment or simulation within the MultiCellDS software. A snapshot contains metadata, which could include user information, software information, experimental setup, and citation information, and a phenotype dataset, which creates mappings of phenotypic measurements with the cellular microenvironment. Snapshots can also reference digital cell being used to create snapshots that come from simulations.

This Master’s Thesis project is a subset of the larger MultiCellDS effort. The results of this project allow for MultiCellDS Digital Snapshots to be converted to the

ISA-Tab data standard, and ISA-Tab data files to be converted to MultiCellDS digital snapshots. The project uses Python code to convert the digital snapshots, which are produced as XML files, to ISA-Tab tab separated text files. All of the information from each file, both data and metadata, is accounted for and transferred to the proper locations in the other file type.

The Python scripts produced this project yield output files that have valid formatting in each data standard and verified contents for all Digital Snapshots currently available in the MultiCellDS Gitlab repository (currently 327 Digital Snapshots)

In the open-source nature of the MultiCellDS, all scripts, spreadsheets, and output files created for this Thesis project will be available on GitHub. The conversion between file types will allow for improved collaboration between researchers by allowing for information to be used in a variety of software packages.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

MultiCellDS/MCDS – MultiCellular Data Standard

ISA-Tab Format/ISA – Investigation/Study/Assay tab-delimited format

DSS – Digital Snapshot

DCL – Digital Cell Line

XML – eXtensible Markup Language

XML Element – Data field in XML file that can contain text or attributes

MCDS Entity – Refers to data field in MCDS file, either an attribute or element, that may be converted to an ISA-Tab Entity

csI. INTRODUCTION

With technological improvements in computing, in silico experimentation has become an increasingly viable option for researchers, rather than solely using in vivo and in vitro models for experimentation (Palsson, 2000). This in silico approach can allow for study of the behavior individual cells, as well as study of tissues and organs. This can allow for cells to be understood as a system that can be described quantitatively.

PhysiCell Simulation is one example of the results of increased viability of in silico experimentation. PhysiCell is an open-source framework that can allow for physics-based, 3D multicellular experimentation (Ghaffarizadeh et al, 2018). The simulation includes models for cell cycling, apoptosis, necrosis, volume exchanges, mechanics, and motility and can feasibly allow for simulation of thousands of cells. Along with simulations of thousands of cells comes the challenge of capturing, storing, and sharing the associated data, as many research groups work with their own data sources, models, and analyses (Macklin, 2019). This can prevent cooperative research between groups, and limit replication studies to verify simulated outcomes. Additionally, much data is not shared between researchers at all (Friedman et al, 2016). The MultiCellular Data Standard attempts to resolve some of these issues.

The MultiCellular Data Standard (MultiCellDS, or MCDS) project aims to create a versatile data standard that allows for the sharing of multicellular experimental, simulation, and clinical data. A common standard can allow for more universal software tools that can be cooperatively developed and used between researchers. MCDS solves issues with storing and sharing data in three main ways – Digital Cell Lines (DCLs), Digital Snapshots (DSSs), and publicly available collectuions of DCLs and DSSs.

Digital Cell Lines focus on the measurements associated with a single cell type, creating phenotype datasets "allowing systematic recording of cell behavior in a single microenvironmental context" (Friedman et al, 2016). The DCL also contains metadata – the information assiciated with who generated the data, and the tools they used to generate the data.

Digital Snapshots focus on storing data with a group of cells at a specific time. Snapshots also contain metadata, but it also contains the data of all cells involved in the simulation (rather than data associated with one cell type). This data can include cell positions, phenotype, and cell types. These data elements are also used for in vivo and in vitro systems, so snapshots can allow for data of many experimental conditions to be shared (Friedman et al, 2016).

The general format of a Digital Snapshot is shown in FIGURE 1 below. As previously mentioned, DSSs contain metadata regarding who generated the data and how they did so. This can include contact information of researchers, citations of relavant studies, and descriptions of the study and data. The bulk of the data contained in the file appears in the Cellular Information and Microenvironment sections of the file. This includes data about the individual cells within a simulation, their properties, characteristics of the environment of the cell, cell locations, and much more.

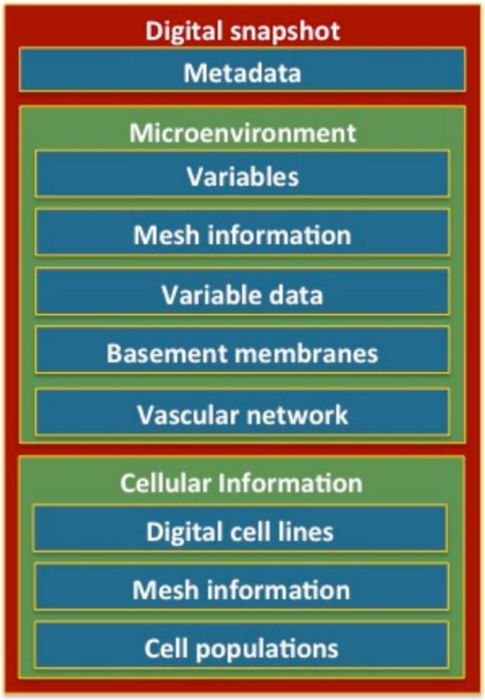


FIGURE 1. ELEMENTS OF A DIGITAL SNAPSHOT

Both Snapshots and Cell Lines are created as eXtensible Markup Language (XML) files that allows for data to be more easily read and understood by humans. XML files have a hierarchical structure, consisting of elements, tags, and attributes. The structure and data types within an XML file are governed by a schema, and the 'address' of data within the file is given by an xPath. Understanding of the structure of an XML file is vital to understanding this project.

In addition to MultiCellDS, Investigation/Study/Assay Tab-Delimited Format, or ISA-Tab format, is a file formatting system that is used to demonstrate all relevant information to multicellular experiments. Like MCDS, ISA-Tab Format defines the formatting of metadata and data that is vital to understanding an experiment and its methods (Rocca-Serra et al, 2009). ISA format dictates that data is stored within three separate text files, an Investigation file, a Study file, and an Assay file. An Investigation file records metadata relating to an institution/researchers and links related Studies under one overall Investigation. A Study file should contain information on the subject that is under study, design of the study, and protocols of the study, as well as give contextualization to Assays within the Study. An Assay file contains information about the tests performed during the Study that produce either qualitative or quantitative data (ISA Abstract Model, n.d.).

Data is aranged hierarchically in ISA-Tab format, but not in the same way as MultiCellDS indicates. Within the ISA files, data is a tab-delimited text file, which is very similar to a CSV file. Files can be opened in Microsoft Excel, and appear essentially as a spreadsheet. However, ISA files do follow a hierarchy of workflow, as well as having a 'nested' structure, where the Investigation file is the 'parent' file to one or more Study files, and the Study file is the 'parent' to one or more Assay files.

While both ISA-Tab and MCDS files contain very similar information, they are formatted in such a way that was previously incompatible. For example, PhysiCell simulation requires input files that are formatted as MCDS XML files. A method to translate the data and formatting of MCDS files to ISA-Tab files, and vice versa. Solving this issue of compatibility would allow for further study of experimental data, and could allow for new knowledge in the area of multicellular biology research.

Adding a layer of complexity, MCDS Digital Cell Lines and Digital Snapshots contain substantially different information. As such, different methods were required for translating their respective data into the ISA format, and vice versa.

Prior to 2020, some previous work had been completed in developing a Python script to convert Digital Cell Lines to the ISA format. However, this script was not fully completed, and did not transfer all of the data contained within Digital Cell Lines to ISA files. The script was also written in Python 2, which is no longer supported in many applications with the development of Python 3. Improvement on this script was needed, and was completed by Connor Burns in April of 2021 (Burns, 2021).

However, prior to 2020, no previous work had been completed on developing a Python script to convert Digital Snapshots to the ISA format. As such, hundreds of Snapshots were stored in a repository without any feasible use in applications where ISA-Tab Format would be required. This Thesis project sought to understand and solve the challenges of conversion of MultiCell Digital Snapshots between data standards and develop a Python script to convert all data contained in the Snapshots between formats.

This project also sought to advance the knowledge of convesion of any ISA-Tab file to the MCDS standard. Currently, the scripts developed can convert the hierarchically structured data of MCDS files between formats. However, major challenges exist to fully automate conversion between ISA-Tab format and MCDS. While the data of the ISA-Tab files is structured, the data contained within the files can vary significantly, depending on the subject, methods, and results of a given Investigation. MCDS files, on the other hand, only contain data that is defined by the MCDS Schema. Because of this issue, this Thesis project was limited in scope. Only files that had been previously converted from MCDS to ISA-Tab format could be translated back to MCDS Digital Snapshots. While all ISA-Tab files cannot be converted to MCDS files yet, this project could serve as a foundation for future steps in solving the challenges of the ISA to MCDS conversion.

II. PROCEDURE

A. MCDS TO ISA-TAB CONVERSION

1. MCDS ENTITY TO ISA-TAB ENTITY MAPPING

As mentioned in the Introduction, some previous work had been completed in converting MCDS DCLs to ISA-Tab format, but it was not completed, and needed updates. To start this project in converting MCDS DSSs to ISA-Tab format, this script was analyzed. The first issue with the original script was that the programming language used to develop the script, Python 2, was outdated. While Python 2, was out of date, Python 3 was a language that is well supported and easy to understand. Python is also a very popular language for data science uses. Additionally, Python has many publicly available libraries that allow for reading and writing to XML and text file formats. As such, it was the language of choice to develop the MCDS DSS to ISA-Tab conversion script. Pycharm Community Edition was chosen as the Python IDE for the project, as it easily allows for pushing script updates to GitHub, package installations, and debugging.

The original script operated by mapping data from a location in the MCDS DCL file to a location in the ISA file within the script. To do this, the script searched for data in the MCDS file by using an xPath for the expected location of the data, and then writing that data to the ISA text file with a corresponding ISA entity. While this original script was incomplete, the method of finding data in the MCDS file and assigning it a location in the ISA file was a method that showed promise. However, individually mapping every possible location within the text of the script seemed an inefficient method. This method could make any future updates to the script much more difficult than necessary.

In order to improve the mapping process, and allow for future updates to the mapping, a method using an Excel database as a mapping reference was proposed. This method would allow for iteration within the script to read the contents of the Excel file and the corresponding mapping. After this initial concept for the script was proposed, further work to plan out the process of developing the scripts was completed.

The general workflow of the project can be seen in FIGURE 2 below, and each step of the chart will be examined throughout the remainder of the METHODS section of this paper.

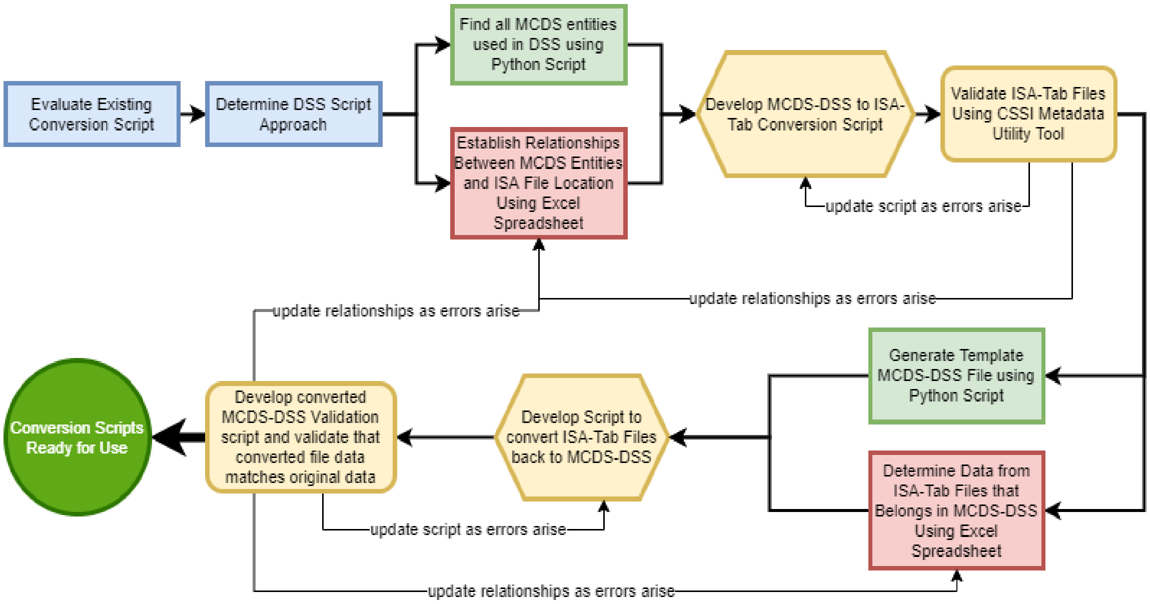

Evaluate Existing Conversion Script
Determine DSS Script Approach
Find all MCDS entities used in DSS using Python Script
Establish Relationships Between MCDS Entities and ISA File Location Using Excel Spreadsheet
Develop MCDS-DSS to ISA-Tab Conversion Script
Validate ISA-Tab Files Using CSSI Metadata Utility Tool
update script as errors arise
update relationships as errors arise
update relationships as errors arise
Generate Template MCDS-DSS File using Python Script
Determine Data from ISA-Tab Files that Belongs in MCDS-DSS Using Excel Spreadsheet
Develop Script to convert ISA-Tab Files back to MCDS-DSS
Develop converted MCDS-DSS Validation script and validate that converted file data matches original data
Conversion Scripts Ready for Use
update script as errors arise
update relationships as errors arise

FIGURE 2. GENERAL PROJECT WORKFLOW

After the original script was examined, and a general method for the new script was proposed, gathering data from the repository of existing Digital Snapshots was essential. In order to convert all of the data from one format to the other, a fundamental understanding of the data contained in the files was required. While there are thousands of possible data locations in a MCDS XML file, not all of these are commonly used in MCDS DSS files. Some data locations may be commonly used in only Digital Cell Lines, while others may only be commonly used in DSS files. In order to determine which MCDS data locations were relevant to DSSs, an additional Python script was developed to find the element name of all tags and attributes in the DSS files that currently existed. This script, ‘Generate_MCDS_Entity_List.py’, used a folder containing all current MCDS DSS files (MCDS_S_0000000001 through MCDS_S_0000000327 at time of writing) as an input, and output an Excel file containing the entity name, xPath, and data type of all data contained within every file.

The script began by iterating through every DSS in the ‘All_Digital_Snapshots’ folder. Within each file, the script then iterated through every element contained in the file. At each data entity in the file, the data type was determined. Possible data types included text elements, which contain only text, attribute elements, which contain text and have attributes, parent elements, which contain no data but have child elements, and attributes, which are contained within a given element and contain additional information about the element. Entries could also appear as closed tags, which do not contain any data, attributes, or child elements. While iterating this data type was appended to a Pandas DataFrame, along with the DSS file name and name of the entity. After iterating through every file and entity, the DataFrame was written to the Excel file. A sample of data contained within the Excel file can be seen in TABLE I below. This sample contains only the first nine MCDS entities, their xPaths, the file name they correspond to, and the type of entity.

TABLE I.

'GENERATE_MCDS_ENTITY_LIST.PY' SCRIPT SAMPLE OUTPUT

MCDS Entity	***xPath***	***File Name***	***Entity Type***
MultiCellDS	/MultiCellDS	<DirEntry 'MCDS_S_0000000001.xml'>	Attribute Element
type	/MultiCellDS[@type]	<DirEntry 'MCDS_S_0000000001.xml'>	Attribute
version	/MultiCellDS[@version]	<DirEntry 'MCDS_S_0000000001.xml'>	Attribute
metadata	/MultiCellDS/metadata	<DirEntry 'MCDS_S_0000000001.xml'>	Parent Element
MultiCellDB	/MultiCellDS/metadata/MultiCellDB	<DirEntry 'MCDS_S_0000000001.xml'>	Parent Element
ID	/MultiCellDS/metadata/MultiCellDB/ID	<DirEntry 'MCDS_S_0000000001.xml'>	Text Element
name	/MultiCellDS/metadata/MultiCellDB/name	<DirEntry 'MCDS_S_0000000001.xml'>	Text Element
description	/MultiCellDS/metadata/description	<DirEntry 'MCDS_S_0000000001.xml'>	Text Element
citation	/MultiCellDS/metadata/citation	<DirEntry 'MCDS_S_0000000001.xml'>	Parent Element

From here, duplicate xPaths were removed from the list in Excel, and every unique xPath found in DSSs was determined. In all, there were 135 unique entities found, with 32 entities being parent elements that contained no data. This left 103 entities, both attributes and text elements, that needed to be mapped to an ISA file location. Not all of these entities appeared in every DSS file. Specifically, the ‘receptor’ element and its attributes only appeared in a few of the 327 files. Additionally, the ‘diagnosis’ tag appeared in every file, but data was only present in some of the elements. To explain, the ‘diagnosis’ element appears 4 times in every file, each with the attribute ‘type’ present. The first occurrence is the primary diagnosis, then secondary, tertiary, and final diagnosis. In all files, a diagnosis, type final, was present. However, the other types of diagnoses occurred inconsistently throughout the files. These differences in the ‘diagnosis’ entity needed to be accounted for in the conversion script.

Once all of the entities appearing in DSSs were determined, it could then be determined where those entities should appear in the ISA format. This proved to be the most time-consuming aspect of the project, as each individual MCDS entity had to be analyzed and categorized based on the information contained.

The Investigation file is grouped into blocks corresponding to the Ontology Source Reference, Investigation, Investigation Publications, Investigation Contacts, Study, Study Design Descriptors, Study Publications, Study Factors, Study Assays, Study Protocols, and Study Contacts (ISA-Tab Format, n.d.). Each of these blocks contains entities that correspond to information about the experiment, and the data contained within each entity is entered into columns to the right of the entity name. In general, most of this Investigation information corresponds to the metadata section of the DSSs. Many of these entities directly correspond to data contained within MCDS files but are given different names. For example, under the ‘Study Contacts’ block, the ISA entity for the institution of the researcher is referred to as ‘Study Person Affiliation’, whereas in MCDS the same data is grouped under the ‘organization-name’ entity. Because of these differences in names, each MCDS entity name had to be compared to existing ISA entities, and the relationship determined.

While each block of data had established entities corresponding to data commonly used in each block, ISA format also allows for comments to be added to the block. These comments must contain data, however, or the file is invalid. The comments allowed for additional information contained within MCDS files to be added to the ISA files. For example, the website of each Study Contact is an entity in MCDS, but not a standard entity in ISA format, so it is mapped to the ISA format as ‘Comment[Study Person URL]’. Conversely, there are several ISA entities that do not have a corresponding MCDS entity, but they are required to create a valid Investigation file. This includes the entire ‘Ontology Source Reference’ block. To resolve this issue, some entities would need to be manually defined in the Excel reference file as ‘Text Entry’ or ‘Script Variables’ that would be defined within the script based on other information in the MCDS file.

There were a few other discrepancies between MCDS entities and ISA-Tab entities that had to be accounted for in the Investigation file mapping. One issue that occurred in several instances involved data from multiple MCDS entities being required for a single ISA entity. This required additional descriptors for these elements in the relationship mapping Excel document. The ‘contacts’ descriptor was added to the mapping spreadsheet to indicate that multiple MCDS file locations associated with individuals involved in the study should be searched to add data to the ISA entity. Each contact was added to a new column within that entity. Similarly, some other entities needed to be combined, but rather than having their own column, they needed to be added to the same ‘cell’ in the Investigation file. For these, an ‘&’ was used to indicate that data needed to be merged. Finally, some ISA data appeared as the name of an element in the MCDS file. This occurred when mapping the roles for each contact. For example, the role of current contact for the study is found as the name of the ‘current-contact’ entity in MCDS. These additional descriptors in the mapping Excel file later made writing the conversion script much easier.

Once the mapping for the Investigation file was complete, MCDS entities that had been mapped to ISA entities were marked off, and the remaining entities were examined to determine whether they belonged in the Study file or the Assay file. While the contents of each of these files were different, the structure of the Study and Assay files was similar to each other, but different than the structure of the Investigation file. Rather than having blocks of data, with a header in the first column, the Study and Assay files were arranged much like a standard table of data, with ISA entities in the first row of the file serving as headers for their respective columns of data.

The Study file is intended to give additional context for the Assay file. This means that the Study file would provide the source of the data in the Assay file, shown as the ‘Source Name’ ISA-Tab entity. Characteristics of the source of data were then given to additionally contextualize the Assay data. This included the location of the study, patient information, and clinical diagnoses for the cells used in the study. Like the

Investigation file, many of the equivalent MultiCellDS entities were located within the metadata section of the Snapshots. However, most of the information for the Study file was located specifically within the ‘cell_origin’ element within the metadata section. Following the characteristics was a ‘Protocol REF’ entity, which shows the protocol used to generate the data in the Assay file. The protocol must also be referenced in the Investigation file in order to create a valid Assay file. The ‘Protocol REF’ was then followed by a ‘Sample Name’, which was used to identify each cell associated with the study. In order to create a valid Study file, there must be a valid ‘Source Name’, ‘Protocol REF’, and ‘Sample Name’ that show the workflow associated with the Study.

The Assay file then contains the information associated with each individual cell from the Snapshots. In the Snapshots this information was located within the ‘cellular_information’ section of the file rather than the ‘metadata’ section. Much like the Study file, the headers of the Assay file show the workflow associated with each cell. The file begins with the ‘Sample Name’ and is followed by a ‘Protocol REF’ defined in the Investigation file. The Assay file is then composed of a few Characteristics that further contextualize the samples, such as the ‘Cell Part’ Characteristic that defines which component of the cell is being analyzed in the experiment. These headers are then followed by Parameters, which contain the quantitative data from measurements. This is where MCDS entities such as ‘orientation’, ‘position’, ‘diameter’, and other measurement-based entities could be mapped. Each Parameter can be further described by additional Characteristics that describe the measurement. Each measurement also had a header to describe the units used for the measurement. Concluding the Assay file and following the Parameters and Characteristics is the name of the Assay and the Snapshot filename.

The Excel file resulting from the mapping of MCDS entities to ISA-Tab entities is too large to copy into the text of this paper. A link to the completed 'MCDS_DSS_2_ISA_Relationships.xlsx' file is found in of APPENDIX I.

After the mapping process was completed, a Python script could then be produced.

2. DEVELOPMENT OF MCDS TO ISA-TAB CONVERSION SCRIPT

As previously mentioned, Pycharm Community Edition was the IDE used for the development of the conversion script. This IDE allowed for multiple Python libraries to be imported with relative ease. The specific packages that were imported for the first conversion script, outside of standard Python libraries, can be found in TABLE II below.

TABLE II.

PYTHON PACKAGES USED IN MCDS TO ISA CONVERSION SCRIPT

Package	***Description of Use in Script***
Pandas	Reads Excel as a DataFrame
LXML	XML data parsing
NumPy	Creation of 2 dimensional arrays for temporary data storage

The MCDS to ISA-Tab conversion script uses the completed mapping Excel document to convert each MCDS DSS from the repository into an Investigation, Study, and Assay file. The outputs will be added to the ‘ISATabOutput’ folder, with a sub-folder containing the ISA files of each individual DSS. In order for the script to function, the mapping document and input folder must be located in the same file path as the conversion script. A general overview of the script can be found in FIGURE 3 below.

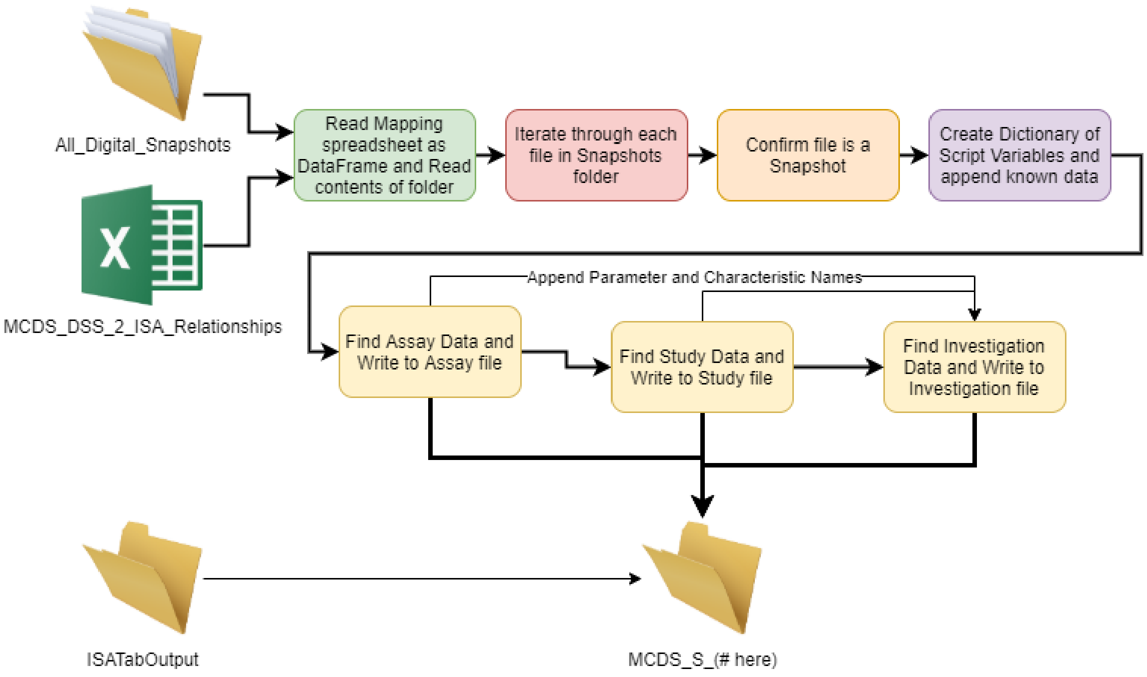

All_Digital_Snapshots
MCDS_DSS_2_ISA_Relationships
Read Mapping spreadsheet as DataFrame and Read contents of folder
Iterate through each file in Snapshots folder
Confirm file is a Snapshot
Create Dictionary of Script Variables and append known data
Append Parameter and Characteristic Names
Find Assay Data and Write to Assay file
Find Study Data and Write to Study file
Find Investigation Data and Write to Investigation file
ISATabOutput
MCDS_S_(# here)

FIGURE 3. MCDS TO ISA CONVERSION SCRIPT FLOW CHART

After establishing the file paths of the relevant folders and files, the script begins by iterating through each MCDS DSS in the 'All_Digital_Snapshots' folder. Each file is first checked that it is an XML file, or it is skipped in the iteration. If the file is an XML file, it is parsed using the LXML submodule called 'ElementTree' or 'etree.' Parsing with this module allows for data to be found from a given xPath within the file, or by finding the name of an element. The XML file is then checked for its version and type, ensuring that the file is a Snapshot (rather than a Digital Cell Line or another type of XML file) and is version '1.0.0', which is the Snapshot version supported by the conversion script.

After the initial setup and verification that the file is a DSS, variables are initialized to pull the name of the DSS and create filenames for the I, S, and A files. An output folder for the given DSS is made within the 'ISATabOutput' folder, if it does not already exist. A dictionary is then initialized for each of the ‘script variables,’ and the filenames and other known data is appended to the dictionary.

After these initial steps are completed, data can begin to be read from the DSS and written to ISA files. While the Investigation file is the ‘parent’ of the Study and Assay files in the ISA-Tab standard, data that belongs in the Assay file is found and written first. This is necessary for the conversion script because the Investigation file contains references to the completed Study and Assay files. If the Investigation file was completed first, it would later have to be modified after the completion of the other files, so writing the files in reverse order created a more efficient script.

In the script, the Assay file mapping was read from the Excel relationships document and imported as a Pandas DataFrame. An empty NumPy array was then created to temporarily store collected data. The width of this array was determined by the number of ISA headers in the DataFrame, and the length of the array was determined by counting the number of cells present in the DSS, as the Assay file includes the quantitative data from each cell.

Data from the DSS is then found by iterating through the rows of the DataFrame, where each row of the DataFrame contains the header name and xPath of the corresponding data. All occurrences of the given xPath are found within the ElementTree of the DSS, and the data from each occurrence is transferred to the corresponding array row. If/elif/else statements are used while iterating through each row of the DataFrame to satisfy differences in the locations of the data within the ElementTree. For example, if the xPath contains an ‘@,’ the corresponding data is found as an attribute in the XML file, so a different function within the LXML module must be used to locate the correct data. In the case of the entities that are attributes, the element containing the attribute must first be found using the ‘findall()’ function, and then the attribute accessed using the ‘get()’ function. Whereas if the xPath of the data does not contain an ‘@’, it is found in the text of an element in the XML file. That data is found in one less step by using the ‘findall().text’ or ‘iter().text’ LXML function.

After iteration through each row of the DataFrame was complete, the array was filled with all of the Assay data contained in the DSS. The script would then iterate through each column and row of the array, writing existing data to the Assay file, separating each data entry by a tab character, and writing a newline character after the last column in each row. While writing the Assay file, all Parameters and Components containing data in the file were appended to a dictionary for later use in the Investigation file.

The Study file was written in a very similar manner as the Assay file. The Study file mapping was read as a Pandas DataFrame that contained the header names and xPaths. Like the assay file, the script iterated through each row of the DataFrame, gathering data from the DSS file. However, the process for writing the Study file differed from the Assay file. Because each cell in every DSS came from the same source, the data contained in each row of the Study file was the same, apart from the Sample Name at the end of each row. Also, the Study file contained fewer entities than the Investigation and Assay files. This meant that an array was not needed to temporarily store data, and a list was used instead. Each entity was searched for in the same manner as the Assay file, and the data found was appended to this list. The data and header for each entity was then written to the Study file line by line, provided that the data existed. Characteristics that contained data were appended to a dictionary for later use in the Investigation file. In each new line, only the Sample Name was changed, and the number of lines in the file corresponded to the number of cells found in the DSS.

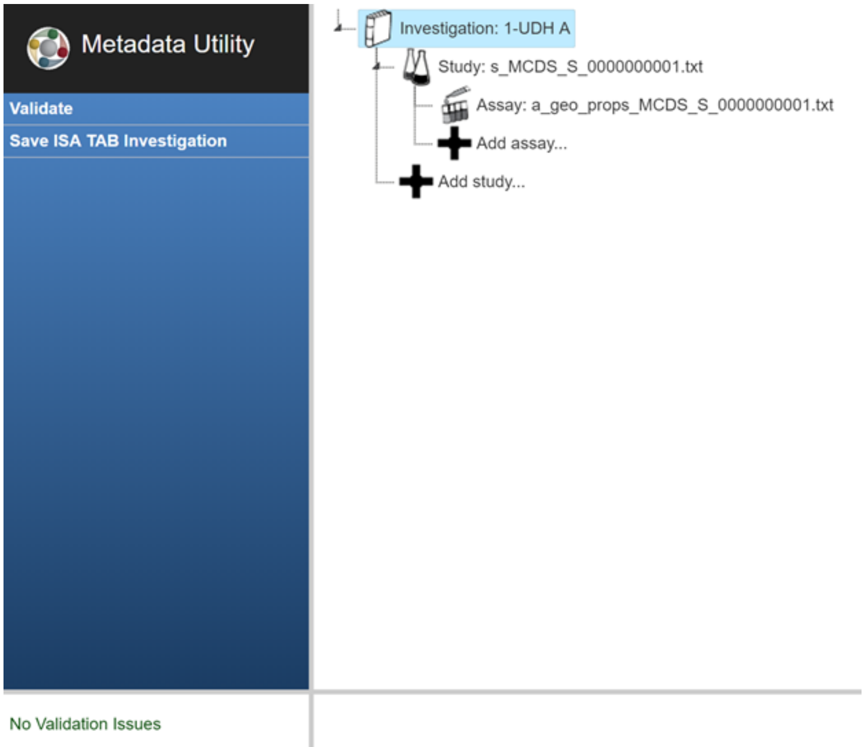
Finally, data for the Investigation file was found and written in a similar manner as the previous two files. As mentioned while explaining the mapping process, there were several different ways data was searched for in the DSS XML file. The method of searching for the data, along with the entity name, xPath, and entity type, was read from the mapping Excel file as a Pandas DataFrame. As in the Assay file, an array was used to temporarily store data before writing it to an Excel file. The script then iterated through each row of the DataFrame. There were several layers of if/elif/else statements to determine where to search for the data in the DSS file, and what to do with the data found. As mentioned in the mapping process, some data needed to be merged from multiple places to be converted to one ISA-Tab entity. The logic statements also determined how many columns of data to write to each ISA block of entities. This section of the script also accounted for all of the ‘script variables’ that had been assigned to dictionaries, including merging all of the Characteristics and Parameters from the Study and Assay files into a single ISA entity. After all of the data was added to the array, the script iterated through each row of the array, writing the associated headers and data on each new line of the Investigation final.

After the Investigation file was written, the script moved onto the next file in the ‘All_Digital_Snapshots’ folder until all available MCDS DSSs were converted to ISA-Tab format.

3. ISA-TAB FILE VALIDATION

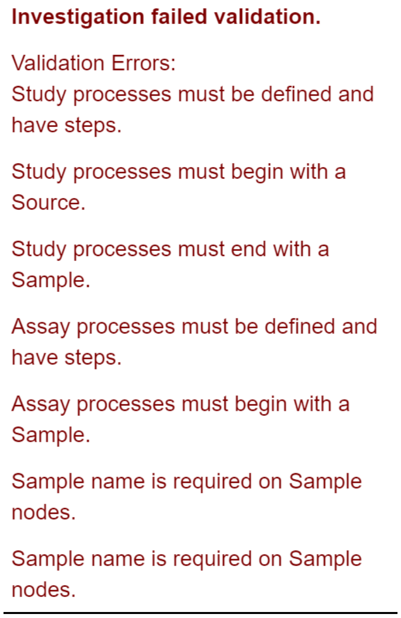
After the conversion script was completed, the ISA-Tab output files needed to be validated. The files were validated using the Center for Strategic Scientific Initiatives (CSSI) Metadata Utility application. This software was obtained by permission of the National Cancer Institute of the National Institute of Health (NIH/NCI). The software was accessed through the CSSI Portal (CSSI Portal, 2020).

For ISA-Tab Format validation, the Investigation file imported in the Metadata Utility, and the software automatically loaded the Study and Assay files associated with the given Snapshot. The filenames of the Study and Assay files had to be correctly listed within the Investigation file for this to work. After the ISA files are loaded, the ‘Validate’ button was clicked. The software then displayed ‘No Validation Issues’, as seen in FIGURE 4, if the ISA files were valid, or displayed any errors found in the files, as seen in FIGURE 5.


Metadata Utility
Validate
Save ISA TAB Investigation
Investigation: 1-UDH A
Study: s_MCDS_S_0000000001.txt
Assay: a_geo_props_MCDS_S_0000000001.txt
Add assay...
Add study...

No Validation Issues

FIGURE 4. ISA FILES PASSING VALIDATION



Investigation failed validation.

Validation Errors:
Study processes must be defined and have steps.

Study processes must begin with a Source.

Study processes must end with a Sample.

Assay processes must be defined and have steps.

Assay processes must begin with a Sample.

Sample name is required on Sample nodes.

Sample name is required on Sample nodes.

FIGURE 5. ISA FILES FAILING VALIDATION

As the validation process required manually loading each file, and only one set of ISA-Tab files could be loaded at one time, validating all 327 sets of ISA-Tab files was not reasonable. To minimize the time required for file validation, while still validating a representative sample of the files, every tenth set of converted ISA-Tab files was validated using the process described above. This sample of files represented all mapped MCDS and ISA-Tab entities. Given that each entity was mapped in the same way, these validated files should represent effectiveness in converting existing MCDS DSS XML files to ISA-Tab text files. A screen clip of the validation of each ISA-Tab file set in the sample is included in APPENDIX III.

B. ISA-TAB TO MCDS CONVERSION

1. ISA-TAB ENTITY TO MCDS ENTITY MAPPING

The process for mapping ISA-Tab entities back to MCDS entities took considerably less time to complete than MCDS to ISA-Tab conversion. This process was much shorter because the ISA-Tab to MCDS mapping was essentially the reverse of the existing mapping, with some minor modifications. The most difficult part of this process was determining a method for creating a converted DSS XML file.

While no previous work for converting DSS files existed, creating a template MCDS XML file proved to be an effective method for ISA-Tab to MCDS Digital Cell Line conversion (Burns, 2021). Rather than producing an XML file from scratch, a more efficient strategy involved removing all of the data from an existing MCDS DSS containing all possible entities. The general process for creating a DSS template can be seen in FIGURE 6 below.

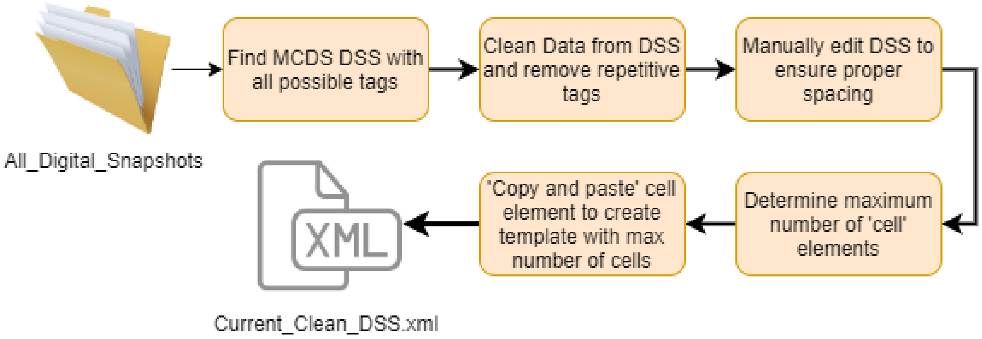
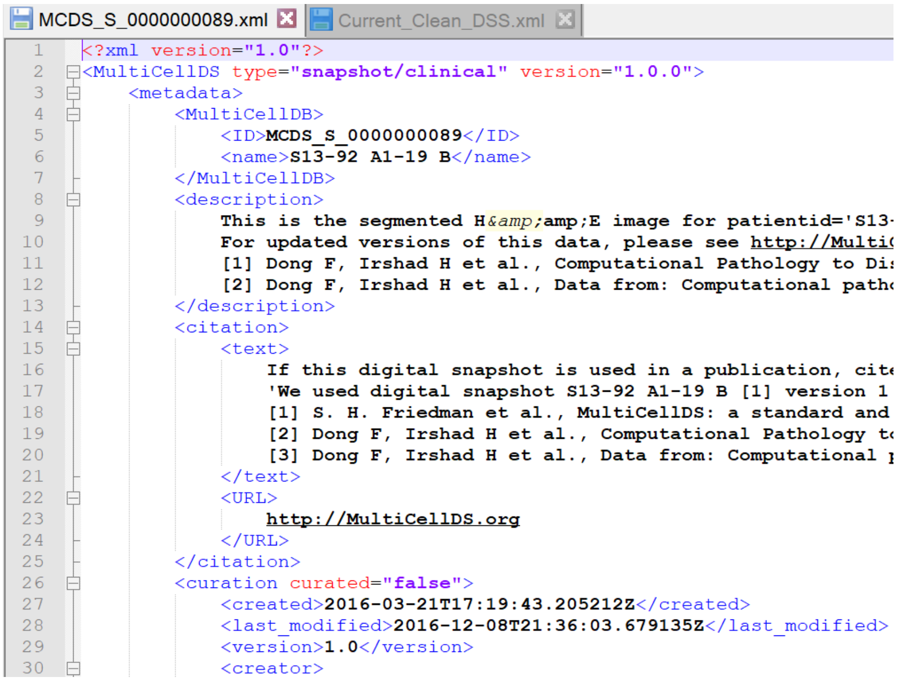

All_Digital_Snapshots
Find MCDS DSS with all possible tags
Clean Data from DSS and remove repetitive tags
Manually edit DSS to ensure proper spacing
Determine maximum number of 'cell' elements
'Copy and paste' cell element to create template with max number of cells
XML
Current_Clean_DSS.xml

FIGURE 6. MCDS DSS TEMPLATE GENERATION WORKFLOW

The ‘Generate_Template_DSS.py’ script was used to create the ‘Current_Clean_DSS.xml’ template Snapshot. A link to this script can be found in APPENDIX I. While all steps to complete the process are found in this script, only small parts of the script were run at any one time, as some manual editing of the template was required between steps.

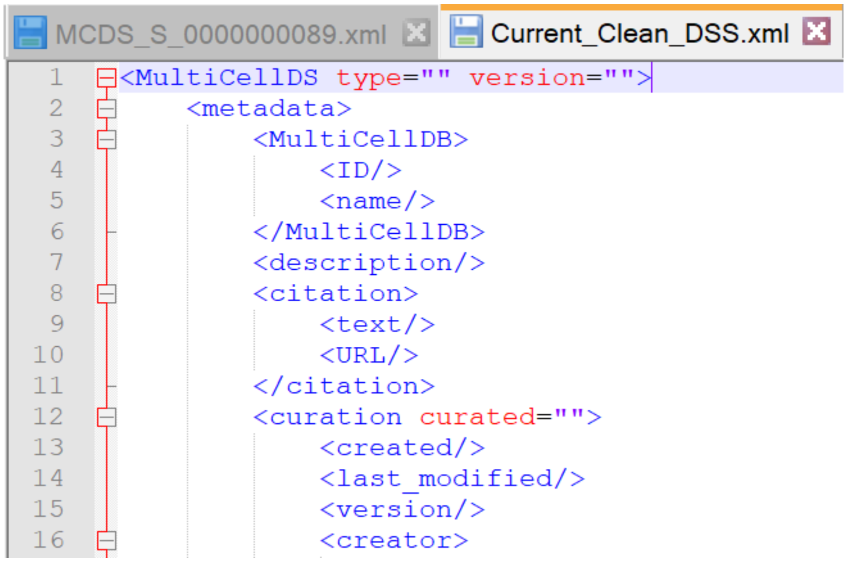
The first step of creating a template DSS was finding an existing DSS that contained all possible entities. This was achieved by using the list of all possible entities form the ‘MCDS_DSS_All_Entities_Sorted.xlsx’ spreadsheet that was originally created while mapping MCDS entities to ISA entities. The script iterated through each DSS in the ‘All_Digital_Snapshots’ folder, appending the entities that were missing from each individual file, as well as appending the filenames of those files which contained every possible entity. From this script, it was determined that DSS ‘MCDS_S_0000000089’ was the first DSS in the folder that met this requirement. This DSS could then be cleaned of all data to produce a template. A section of MCDS_S_0000000089 prior to cleaning can be seen in FIGURE 7 below.



```
<?xml version="1.0"?>
<MultiCellDS type="snapshot/clinical" version="1.0.0">
    <metadata>
        <MultiCellDB>
            <ID>MCDS_S_0000000089</ID>
            <name>S13-92 A1-19 B</name>
        </MultiCellDB>
        <description>
            This is the segmented H&amp;amp;E image for patientid='S13-
            For updated versions of this data, please see http://MultiC
            [1] Dong F, Irshad H et al., Computational Pathology to Dis
            [2] Dong F, Irshad H et al., Data from: Computational patho
        </description>
        <citation>
            <text>
                If this digital snapshot is used in a publication, cite
                'We used digital snapshot S13-92 A1-19 B [1] version 1
                [1] S. H. Friedman et al., MultiCellDS: a standard and
                [2] Dong F, Irshad H et al., Computational Pathology to
                [3] Dong F, Irshad H et al., Data from: Computational p
            </text>
            <URL>
                http://MultiCellDS.org
            </URL>
        </citation>
        <curation curated="false">
            <created>2016-03-21T17:19:43.205212Z</created>
            <last_modified>2016-12-08T21:36:03.679135Z</last_modified>
            <version>1.0</version>
            <creator>
```

FIGURE 7. MCDS_S_0000000089 PRIOR TO DATA CLEANING

The next portion of the script parsed the XML file using the LXML ElementTree and iterated through each element, removing all data contained as element text, as well as removing any data contained within attributes. After cleaning data, the resulting XML file was a bit disorganized, as all tab and newline characters contained in each element were deleted. After manually adding tab and newline characters to match the white space of the original DSS, the cleaned DSS can be seen in FIGURE 8 below.



```xml
<MultiCellDS type="" version="">
    <metadata>
        <MultiCellDB>
            <ID/>
            <name/>
        </MultiCellDB>
        <description/>
        <citation>
            <text/>
            <URL/>
        </citation>
        <curation curated="">
            <created/>
            <last_modified/>
            <version/>
            <creator>
```

FIGURE 8. MCDS_S_0000000089 FOLLOWING DATA CLEANING

After a ‘rough draft’ of the template was completed, the template was edited to create a finished template. This involved iterating through each original DSS to determine the maximum number of cells found in a DSS. This was determined to be 1132 cells. This meant that the ‘cell’ element and its children should be repeated 1132 times. This process was achieved by manually copying the cell element and its children into a text file. This text file was then read by the script and rewritten an additional 1131 times to the end of the text file. This text was then manually copied and pasted back into the template DSS, completing the ‘Current_Clean_DSS.xml’ file.

Creating a template corresponding to the maximum DSS file size may seem counterintuitive, as the template is significantly larger than most DSSs, but this allows for a simpler conversion script. When converting ISA-Tab data back to a DSS, the LXML function ‘remove(element)’ allow for elements and their children to be removed in one step. Meanwhile, building new elements in the ElementTree requires lines of code and additional LXML functions to be used.

2. DEVELOPMENT OF ISA-TAB TO MCDS CONVERSION SCRIPT

After completion of the template DSS file, the mapping process for the ISA-Tab to MCDS conversion script could then be completed. The mapping process was straightforward, as it was essentially the same as the MCDS to ISA-Tab mapping. The main difference between the 'forward' and 'reverse' mapping was that some data found in the ISA-Tab files was ignored in the conversion script, as it did not originate from the DSS file itself, but rather was defined within the MCDS to ISA script. This included all 'Text Entry' and 'Script Variable' entities from the original mapping. Other than skipping these elements, no other major changes were made to the original mapping.

After the small changes were made to complete the mapping process, the ISA to MCDS conversion script could then be started. It is important to emphasize the limitations of this script. The ISA to MCDS conversion script can only revert ISA files generated from the MCDS to ISA conversion script back to their original Snapshot. The script cannot convert every ISA-Tab file set to a MCDS DSS at this time. There are multiple challenges that prevent this, as will be elaborated on in the CONCLUSIONS AND RECOMMENDATIONS section.

The workflow for the ISA to MCDS DSS conversion script can be seen in FIGURE 9 below. The link to the complete 'ISA_2_MCDS_DSS.py' script can be found in APPENDIX I.

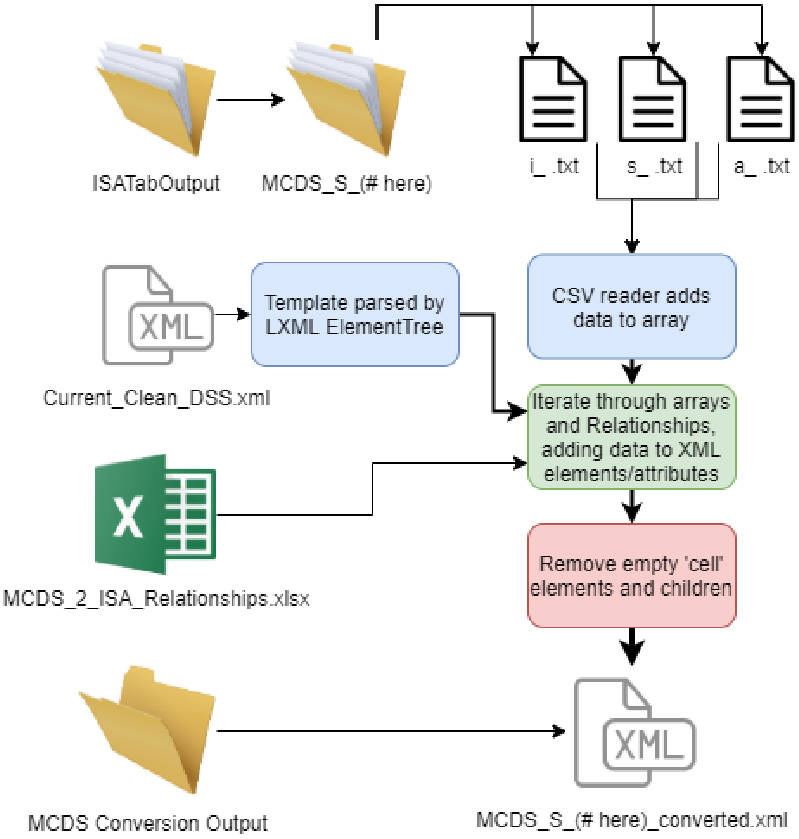
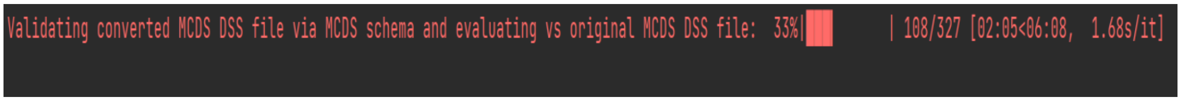

ISATabOutput
MCDS_S_(# here)
i_.txt
s_.txt
a_.txt
Current_Clean_DSS.xml
Template parsed by LXML ElementTree
CSV reader adds data to array
Iterate through arrays and Relationships, adding data to XML elements/attributes
MCDS_2_ISA_Relationships.xlsx
Remove empty 'cell' elements and children
MCDS Conversion Output
MCDS_S_(# here)_converted.xml

FIGURE 9. ISA TO MCDS CONVERSION SCRIPT FLOW CHART

The ISA to MCDS conversion script converts one set of input ISA-Tab files into a MCDS Digital Snapshot output that is identical to the original Snapshot file. The ‘reverse’ script uses the same Python packages as the ‘forward’ script, with the addition of the TQDM package. TQDM was used to track the progress of the script, as the run time associated with this script was much greater than the previous script. This progress bar also showed the average time taken to complete each file conversion. The result of the TQDM package can be seen in FIGURE 10 below. The full list of packages used in the script is found in TABLE III below.



```
Validating converted MCDS DSS file via MCDS schema and evaluating vs original MCDS DSS file:  33%|███      | 108/327 [02:05<06:08,  1.68s/it]
```

FIGURE 10. TQDM PROGRESS BAR

TABLE III.

PYTHON PACKAGES USED IN ISA TO MCDS CONVERSION SCRIPT

Package	***Description of Use in Script***
Pandas	Reads Excel as a DataFrame
LXML	XML data parsing
NumPy	Creation of 2 dimensional arrays for temporary data storage
TQDM	Script Progress Bar

As shown in FIGURE 9, the script first reads the set of ISA-Tab files using a CSV reader. Although ISA files are written as text files, the data they contain can be read more easily by the CSV reader, which reads each ‘cell’ of data in the ISA files, than the text file reader, which reads the text line by line. Using a CSV reader thus removes extra steps of splitting each line to read each ‘cell’ individually. The data from each file is then added to a corresponding array for temporary storage

After the data is added to arrays, the mapping Excel spreadsheet is read as a Pandas DataFrame. The script begins transferring data by iterating through the rows of the Investigation file array. The ISA entity in each row is searched in the DataFrame, and the xPath and corresponding MCDS entity name is found. Using the ‘find(xPath).text’ function, the text of an element in the ElementTree is set equal to the data contained within the ‘cell’ of the Investigation array. If the MCDS entity is an array, the ‘find(xPath).set(attribute, data)’ function is used instead, where ‘data’ is the data contained within the ‘cell’ of the Investigation array.

The process for iteration through the Study and Assay is very similar, but the script iterates through each column of the arrays rather than the rows, as each column corresponds to a MCDS entity in these files. In the Study file, only the first row of data is read, however, as each line of the file is identical apart from the ‘Sample Name’ column, which is not a MCDS entity. For each column, the ISA header is searched in the Pandas DataFrame, and the corresponding MCDS entity names and xPaths are found. The Study data is then added to the ElementTree using the ‘findall(xPath).text’ function, or by the ‘findall(xPath).set(attribute, data)’ function if the MCDS entity is an attribute.

The process of adding the Assay file data to the DSS is identical to the process for adding Study data to the DSS, except the script also iterates through each row of the Assay data array. As previously mentioned, each row of the Assay file corresponds to each ‘cell’ element in the DSS, so each row contains different data, whereas the Study file contains the same data in each row.

After all ISA data is added to the ElementTree, the script iterates through each cell element to check if it contains data. ‘Cell’ elements not containing data are removed using the ‘remove(element)’ function.

The ISA data can then be written to a MCDS DSS XML file named ‘MCDS_S_(#here)_converted.xml.’ This file should be an identical copy to the corresponding original DSS.

3. CONVERTED MCDS DSS VALIDATION

After all sets of ISA files were converted to MCDS XML files, a validation script was developed to validate that the XML files produced satisfied the requirements of MCDS and to verify that all data was successfully transferred. An existing MCDS Digital Cell Line validation script had been previously completed (Burns, 2021), but this script used a Python package that was very inefficient when used with Digital Snapshots, as Snapshots contain up to 200 times the number of entities of Cell Lines. This meant that a validation script specifically for Snapshots was needed. The Python packages used in this script can be seen in TABLE IV. The general workflow of the script can be seen in

TABLE IV.

PYTHON PACKAGES USED IN CONVERTED SNAPSHOT VALIDATION SCRIPT

Package	***Description of Use in Script***
Pandas	Reads Excel as a DataFrame
LXML	XML data parsing
TQDM	Script Progress Bar
XMLSchema	Validation of XML files using XML Schemas (XSDs)

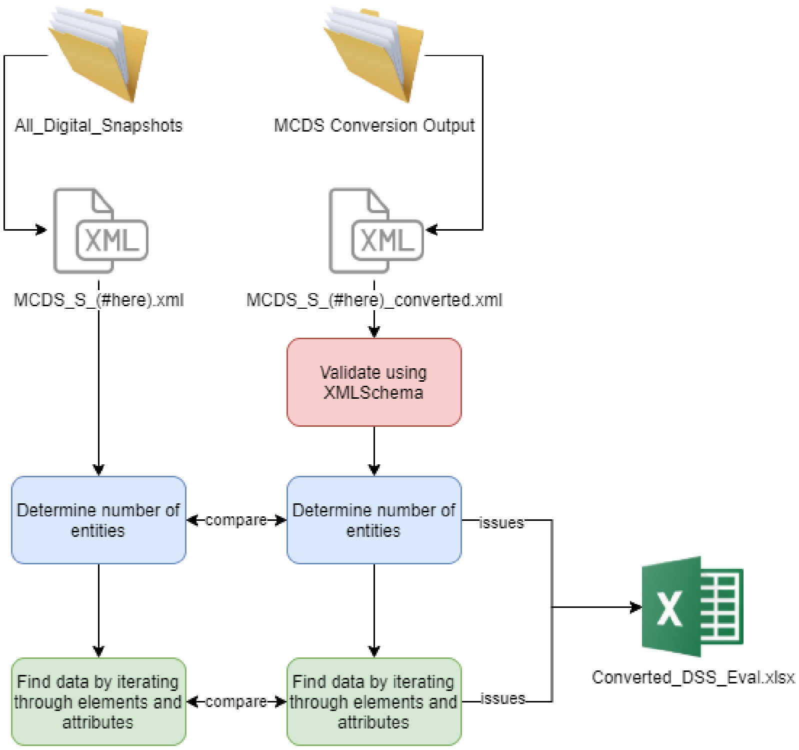

All_Digital_Snapshots
MCDS Conversion Output
XML
XML
MCDS_S_(#here).xml
MCDS_S_(#here)_converted.xml
Validate using XMLSchema
Determine number of entities
compare
Determine number of entities
issues
Converted_DSS_Eval.xlsx
Find data by iterating through elements and attributes
compare
Find data by iterating through elements and attributes
issues

FIGURE 11. ‘CONVERTED_SNAPSHOT_VALIDATION.PY’ WORKFLOW

To begin the script, the original DSS was found in the ‘All_Digital_Snapshots’ folder, while the converted script was found in the ‘MCDS Conversion Output’ folder, which was the output of the ‘ISA_2_MCDS_DSS.py’ script. As shown in FIGURE 11, the first step for validating and verifying the data in the MCDS DSS files was to use the XMLSchema package. After assigning the path of the MCDS Schema, this package allowed the script to ensure that the formatting of entities in the converted DSS corresponded to the expected formatting of a DSS as defined by the data standard by using the ‘is_valid()’ function.

After the file was validated, the data within the XML file could be examined. The first step to ensure that all data was transferred to the converted DSS was comparing the number of MCDS entities in the original and converted file ElementTrees. This served as an initial check to locate any missing data, as entities that were missing in the converted DSS were appended to a Pandas DataFrame. If the number of entities was the same in each file, the data within these entities could then be compared to ensure the files were identical.

After this initial check, the script iterated through each element and attribute in the respective ElementTrees. The script checked that the name of each element/attribute matched and checked that the text contained within each element/attribute matched. If any name or data did not match, the file name, along with the type and location of the error in each file, was appended to the DataFrame. After completing iteration through each individual file, the number of errors in the file was counted and also appended to the DataFrame. After completing iteration through all DSS files, the DataFrame was written to ‘Converted_DSS_Eval.xlsx’ for review.

This DSS validation and verification script also served as a validation and verification of all other scripts produced during the course of this Thesis Project. While the formatting of the ISA-Tab files could be validated using the CSSI Metadata Utility, the contents of each file could not yet be verified, as no previous DSS files had been converted to the ISA-Tab format. However, given that the ISA-Tab files produced in the Project were later reverted to DSS files, comparisons between the original and converted files showed where data may be missing. Edits were then made to the ‘MCDS_DSS_2_ISA_Relationships.xlsx’, ‘MCDS_DSS_2_ISA.py’, and

‘ISA_2_MCDS_DSS.py’ files in order to correct issues causing the data to be missing in the converted files.

After errors were mitigated in each script, the project was then pushed to GitHub to allow for future use of the conversion scripts to those that require it. The link to the GitHub repository, and each individual file within the repository, can be found in TABLE VIII. of APPENDIX I.

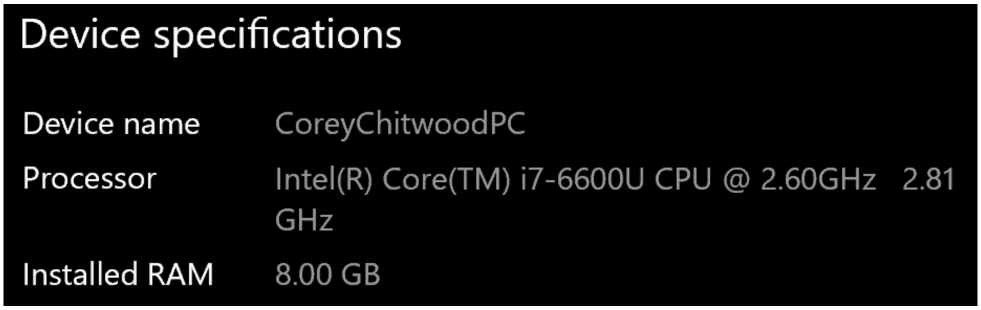
III. RESULTS AND DISCUSSION

While executing the conversion scripts, as well as the DSS validation and verification script, script run times were recorded in order to quantify their efficiencies. These run times are recorded in TABLE V. It is important to note the large differences in the run times for the 'ISA_2_MCDS_DSS.py' script compared to the other two scripts. This script included a loop that located a very large amount of Assay file data and assigned it to the output XML file. This appeared to be the main reason this script required more time to execute. Also of note, the Python scripts were executed on a PC with the Technical Specifications seen in FIGURE 12. Running the scripts on another computer with different RAM and a different processor may yield different run times.

TABLE V.

SCRIPT RUN TIMES

Script	***Total Run Time (327 files)***	***Average Run Time per file (sec/file)***
MCDS_DSS_2_ISA.py	2:46	0.51
ISA_2_MCDS_DSS.py	16:21:23	180.07
Converted_Snapshot_Validation.py	6:09	1.13



Device specifications

Device name	CoreyChitwoodPC
Processor	Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz 2.81 GHz
Installed RAM	8.00 GB

FIGURE 12. DEVICE SPECIFICATIONS OF PC USED TO EXECUTE SCRIPTS

After the ‘MCDS_DSS_2_ISA.py’ script was executed, the output ISA files were validated as described in the II. *PROCEDURE*. As described, approximately every tenth file was validated using the CSSI Metadata Utility application, with the validation results being shown in TABLE VI below. Screen clips of the validation results within the CSSI Metadata Utility application corresponding to each of these ISA-Tab file sets can be found in APPENDIX III.

Of note, TABLE VI only shows the validation results for the finalized ‘MCDS_DSS_2_ISA.py’ script output ISA files. Multiple initial iterations of the script produced invalid ISA-Tab files because of issues in both the script and mapping. These issues were resolved in the development process, and valid ISA-Tab files are now produced by the script.

TABLE VI.

ISA-TAB FILE VALIDATION

MCDS File Name	***ISA-Tab Files Validated?***
MCDS_S_0000000001	yes
MCDS_S_0000000010	yes
MCDS_S_0000000020	yes
MCDS_S_0000000030	yes

MCDS_S_0000000040	yes
MCDS_S_0000000050	yes
MCDS_S_0000000060	yes
MCDS_S_0000000070	yes
MCDS_S_0000000080	yes
MCDS_S_0000000090	yes
MCDS_S_0000000100	yes
MCDS_S_0000000110	yes
MCDS_S_0000000120	yes
MCDS_S_0000000130	yes
MCDS_S_0000000140	yes
MCDS_S_0000000150	yes
MCDS_S_0000000160	yes
MCDS_S_0000000170	yes
MCDS_S_0000000180	yes
MCDS_S_0000000190	yes
MCDS_S_0000000200	yes
MCDS_S_0000000210	yes
MCDS_S_0000000220	yes
MCDS_S_0000000230	yes
MCDS_S_0000000240	yes
MCDS_S_0000000250	yes
MCDS_S_0000000260	yes
MCDS_S_0000000270	yes
MCDS_S_0000000280	yes
MCDS_S_0000000290	yes
MCDS_S_0000000300	yes
MCDS_S_0000000310	yes
MCDS_S_0000000320	yes
MCDS_S_0000000327	yes

The ISA-Tab file sets that were output by the ‘MCDS_DSS_2_ISA.py’ script were then converted back to DSS files using the ‘ISA_2_MCDS_DSS.py’ script. These converted DSS files were then validated, and their data verified, by the ‘Converted_Snapshot_Validation.py’ script. A sample of the output Excel file from the validation script can be seen below in TABLE VII. The full version of this file can be seen in TABLE IX of APPENDIX II.

TABLE VII.

SAMPLE OF 'CONVERTED_SNAPSHOT_VALIDATION.PY' OUTPUT

DSS Filename	***Passed MCDS Validation***	***Total Data Entities in Original File***	***Total Data Entities in Converted File***	***Entity Count Equal?***	***Num of Issues in Converted File***
MCDS_S_0000000001	TRUE	28887	28887	yes	1
MCDS_S_0000000002	TRUE	13201	13201	yes	1
MCDS_S_0000000003	TRUE	23321	23321	yes	1
MCDS_S_0000000004	TRUE	17985	17985	yes	1
MCDS_S_0000000005	TRUE	17295	17295	yes	1

As shown, all converted DSS XML files passed XMLSchema validation. TABLE VII. also shows that there was only one issue in each file when verifying the data compared to the original DSS. Upon further investigation, each file contained the same error, and this error was trivial. The issue occurred within the text of the ‘notes’ element of each file (xPath - metadata/data_origins/data_origin/notes). The text within the ‘notes’ element of the converted DSS had a quotation mark that was misplaced compared to the original, as shown below. The quotation marks in question are underlined in the text of each element.

Original DSS Notes element:

<notes>To access the original image files, download the folder "Data: Original Images" from http://datadryad.org/resource/doi:10.5061/dryad.pv85m</notes>

Converted DSS Notes element:

<notes>To access the original image files, download the folder Data: Original Images" from http://datadryad.org/resource/doi:10.5061/dryad.pv85m"</notes>

The exact cause of this small error could not be definitively determined, but it is presumed to have occurred somewhere within the ‘ISA_2_MCDS_DSS.py’ script, as each Investigation file contained text identical to the original DSS. Other text elements with quotation marks did not have this issue. However, given the trivial nature of the error, combined with the thousands of other entities in each converted DSS file that contained data identical to the original DSS, the error can be overlooked, and the conversion process confirmed as a success.

IV. CONCLUSIONS AND RECOMMENDATIONS

As shown in the RESULTS AND DISCUSSIONS section of this Thesis, the conversion scripts have shown to be successful in achieving the goals of this Project. MultiCellDS Digital Snapshots were used to create valid ISA-Tab files, and those ISA-Tab files were used to create a Snapshot identical to the original. All 327 ISA-Tab output file sets were validated using the CSSI Metadata Utility application, while all 327 converted MCDS DSS files were validated with the MCDS Schema using the validation/verification script developed during this Project. No data was lost during the conversion process, as also confirmed by the validation/verification script.

As additional DSS files are produced and added to the public repository, the conversion scripts and mapping document will require updating. However, the scripts and mapping are completed in such a way that will allow for updates without creating an entirely new conversion script. Mapping between the DSS and Assay file would be the most likely to require updating. All current DSS files are limited to Cell Geometrical Property measurements, despite many more ‘cellular_information’ sub-elements existing in the MultiCell Data Standard. This could require additional Assay files to be created for each individual DSS, as is required in DCL conversion (Burns, 2021).

This Thesis Project also lays groundwork for future automated conversion of ANY ISA-Tab file to MCDS, rather than a only creating an identical copy of an existing

DSS. Currently, new ISA-Tab files would have to be edited within the CSSI Metadata Utility application. The names of ISA entities could be manually changed to match names of ISA entities that were mapped to MCDS during this project, and a valid MCDS DSS could theoretically be produced. However, this would be a difficult process, and would require substantial user edits. Future work could be completed to expand mapping between ISA and MCDS entities to help automate this process.

While expanding mapping ISA and MCDS entities would allow for conversion of more files, this process may require additional scripts and mapping documents. Complicating the conversion is that ISA-Tab format does not have a standardized list of entities that are possible. ISA-Tab is merely a format for data and metadata that is recorded and named by each individual user. Many ISA-Tab users may refer to the same data with different names for the measurements as a result of this lack of universal standard. Successfully mapping all ISA-Tab files to MCDS would require extensive examination of ISA-Tab file sets from many different users. This data could then be used to associate varying ISA-Tab entity names to the MCDS entity that they correspond to.

Another area of the Thesis Project that could be improved upon in the future is the runtime for the ‘ISA_2_MCDS_DSS.py’ script. This script took several minutes to convert each ISA file set to a DSS, and over 16 hours to convert all 327 file sets. The vast majority of this time was lost in the conversion of Assay file data, as this process required iteration through 40 columns and hundreds of rows of data for each file. This process will never be instantaneous, given the large amount of data involved, but there are several potential methods to speed up this portion of the script. One potential solution is using Python list comprehension rather than For-loops, as well as limiting the use of dot notation in the scripts. These are two well-known sources of runtime issues in Python. Also, additional Python packages could allow for portions of the script to be compiled immediately, rather than having the code executed line by line. This option was briefly explored, but initial research, trial, and error indicated that these types of packages may not work well with Pandas DataFrames, which are used throughout each script. Another potential improvement would be the parallel processing of independent functions within the Python scripts. This would allow these sections of code to be executed at the same time, rather than sequentially, which would reduce the overall runtime of the conversion script. However, this method of executing the script would require use of a machine with multiple processors, which would not have been possible on the PC used in the development of the current scripts. Multithreading would be another option, similar to parallel processing, that could be explored for improving runtimes on single processor machines.

A final recommendation for the conversion scripts would be incorporating command line interface. The current versions of the scripts have predetermined file and folder names within the text of the script. This works well if all files to be converted are in the same folder and have the same folder names as those in the script. However, many users may not have files organized in the same way they were organized during the production of these scripts. Incorporating command line interface would allow for the user to further interact with the script to define their own file paths without having to manually edit the script to alter the current file paths.

REFERENCES CITED

Burns, C. J. (2021). Automated Conversion of MultiCellDS Digital Cell Lines and ISA-Tab filesets. *Electronic Thesis and Dissertations*. Retrieved from https://ir.library.louisville.edu/etd/3443/

CSSI Portal. (2020, November). Retrieved from NIH National Cancer Institute Center for Strategic Scientific Initiatives: https://cssi-dcc.nci.nih.gov/cssiportal/

Friedman, S. H et al. (2016). MultiCellDS: a standard and a community for sharing multicellular data.

Ghaffarizadeh, A., Heiland, R., Friedman, S. H., Mumenthaler, S. M., & Macklin, P. (2018). PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. *PLoO Computational Biology*. doi:https://doi.org/10.1371/journal.pcbi.1005991

ISA Abstract Model. (n.d.). Retrieved from ISA Model and Serialization Specifications: https://isa-specs.readthedocs.io/en/latest/isamodel.html

ISA-Tab Format. (n.d.). Retrieved from ISA Model and Serialization Specifications: https://isa-specs.readthedocs.io/en/latest/isatab.html

Macklin, P. (2019). Key challenges facing data-driven multicellular systems biology. *GigaScience, Volume 8*(Issue 10). doi:https://doi.org/10.1093/gigascience/giz127

Palsson, B. (2000). The challenges of in silico biology. *Nature Biotechnology*. doi:https://doi.org/10.1038/81125

Rocca-Serra, P. et al. (2009). Specification Document: ISA-TAB 1.0. *Zenodo*. doi:http://doi.org/10.5281/zenodo.161355

APPENDIX I.

TABLE VIII.

PROJECT FILES AND LOCATIONS

File	***Description***	***Link***
MCDS_DSS_2_ISA.py	Python script used to convert MCDS DSS XML files to an ISA-Tab file set	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/MCDS_DSS_2_ISA.py
ISA_2_MCDS_DSS.py	Python script used to convert ISA-Tab file sets to MCDS DSS XML files	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/ISA_2_MCDS_DSS.py
Generate_MCDS_Entity_List.py	Python script used to determine all possible MCDS DSS entities used	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/Generate_MCDS_Entity_List.py
Converted_Snapshot_Validation.py	Python script used to validate converted MCDS DSS files using the MCDS XML Schema and to verify that data in converted MCDS DSS XML files matches the original MCDS DSS XML files	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/Converted_Snapshot_Validation.py
Generatte_Template_DSS.py	Python script used to create a MCDS DSS with no data	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/Generate_Template_DSS.py
MCDS_DSS_2_ISA_Relationships.xlsx	Excel file containing all mapping between MCDS and ISA-Tab entities	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/MCDS_DSS_2_ISA_Relationships.xlsx

MCDS_DSS_All_Entities_Sorted.xlsx	Excel file containing the sorted output of the ‘Generate_MCDS_Entitiy_List.py’ script	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/MCDS_DSS_All_Entities_Sorted.xlsx
Converted_DSS_Eval.xlsx	Excel File containing the ‘Converted_Snapshot_Validation.py’ script output	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/Converted_DSS_Eval.xlsx
All_Digital_Snapshots Folder	Folder containing all MCDS DSS files	https://github.com/rheiland/mcds2isa/tree/DSS_ISA_Conversion_v1.0/All_Digital_Snapshots
ISATabOutput Folder	Folder containing all sets of ISA-Tab files created by the ‘MDCS_DSS_2_ISA.py’ conversion script	https://github.com/rheiland/mcds2isa/tree/DSS_ISA_Conversion_v1.0/ISATabOutput
MCDS Conversion Output Folder	Folder containing all converted MCDS DSS files created by the ‘ISA_2_MCDS_DSS.py’ conversion script	https://github.com/rheiland/mcds2isa/tree/DSS_ISA_Conversion_v1.0/MCDS%20Conversion%20Output
Current_Clean_DSS.xml	MCDS DSS XML file that has had all data removed to be used as a template for the ‘ISA_2_MCDS_DSS.py’ conversion script	https://github.com/rheiland/mcds2isa/blob/DSS_ISA_Conversion_v1.0/Current_Clean_DSS.xml
MultiCellDS-transitions-v1.0-v1.0.0 Folder	Folder containing all MCDS XML Schemas	https://gitlab.com/MultiCellDS/MultiCellDS/-/tree/master/v1.0/v1.0.0

APPENDIX II.

TABLE IX.

FULL MCDS VALIDATION SCRIPT OUTPUT

DSS Filename	***Passed MCDS Validation***	***Total Data Entities in Original File***	***Total Data Entities in Converted File***	***Entity Count Equal?***	***Num of Issues in Converted File***
MCDS_S_0000000001	TRUE	28887	28887	yes	1
MCDS_S_0000000002	TRUE	13201	13201	yes	1
MCDS_S_0000000003	TRUE	23321	23321	yes	1
MCDS_S_0000000004	TRUE	17985	17985	yes	1
MCDS_S_0000000005	TRUE	17295	17295	yes	1
MCDS_S_0000000006	TRUE	15731	15731	yes	1
MCDS_S_0000000007	TRUE	19227	19227	yes	1
MCDS_S_0000000008	TRUE	13799	13799	yes	1
MCDS_S_0000000009	TRUE	23643	23643	yes	1
MCDS_S_0000000010	TRUE	18445	18445	yes	1
MCDS_S_0000000011	TRUE	42181	42181	yes	1
MCDS_S_0000000012	TRUE	41767	41767	yes	1
MCDS_S_0000000013	TRUE	19273	19273	yes	1

MCDS_S_0000000014	TRUE	29439	29439	yes	1
MCDS_S_0000000015	TRUE	24701	24701	yes	1
MCDS_S_0000000016	TRUE	29623	29623	yes	1
MCDS_S_0000000017	TRUE	33625	33625	yes	1
MCDS_S_0000000018	TRUE	22907	22907	yes	1
MCDS_S_0000000019	TRUE	41031	41031	yes	1
MCDS_S_0000000020	TRUE	27553	27553	yes	1
MCDS_S_0000000021	TRUE	47701	47701	yes	1
MCDS_S_0000000022	TRUE	33119	33119	yes	1
MCDS_S_0000000023	TRUE	32659	32659	yes	1
MCDS_S_0000000024	TRUE	22309	22309	yes	1
MCDS_S_0000000025	TRUE	18537	18537	yes	1
MCDS_S_0000000026	TRUE	48345	48345	yes	1
MCDS_S_0000000027	TRUE	39927	39927	yes	1
MCDS_S_0000000028	TRUE	28887	28887	yes	1
MCDS_S_0000000029	TRUE	27093	27093	yes	1
MCDS_S_0000000030	TRUE	17617	17617	yes	1
MCDS_S_0000000031	TRUE	20699	20699	yes	1
MCDS_S_0000000032	TRUE	34085	34085	yes	1
MCDS_S_0000000033	TRUE	41721	41721	yes	1

MCDS_S_0000000034	TRUE	38317	38317	yes	1
MCDS_S_0000000035	TRUE	44113	44113	yes	1
MCDS_S_0000000036	TRUE	36615	36615	yes	1
MCDS_S_0000000037	TRUE	38225	38225	yes	1
MCDS_S_0000000038	TRUE	41905	41905	yes	1
MCDS_S_0000000039	TRUE	34867	34867	yes	1
MCDS_S_0000000040	TRUE	37397	37397	yes	1
MCDS_S_0000000041	TRUE	28892	28892	yes	1
MCDS_S_0000000042	TRUE	13206	13206	yes	1
MCDS_S_0000000043	TRUE	23326	23326	yes	1
MCDS_S_0000000044	TRUE	17990	17990	yes	1
MCDS_S_0000000045	TRUE	17300	17300	yes	1
MCDS_S_0000000046	TRUE	15736	15736	yes	1
MCDS_S_0000000047	TRUE	19232	19232	yes	1
MCDS_S_0000000048	TRUE	13804	13804	yes	1
MCDS_S_0000000049	TRUE	23648	23648	yes	1
MCDS_S_0000000050	TRUE	18450	18450	yes	1
MCDS_S_0000000051	TRUE	13344	13344	yes	1
MCDS_S_0000000052	TRUE	23970	23970	yes	1
MCDS_S_0000000053	TRUE	15506	15506	yes	1

MCDS_S_0000000054	TRUE	8882	8882	yes	1
MCDS_S_0000000055	TRUE	27788	27788	yes	1
MCDS_S_0000000056	TRUE	13712	13712	yes	1
MCDS_S_0000000057	TRUE	18404	18404	yes	1
MCDS_S_0000000058	TRUE	5800	5800	yes	1
MCDS_S_0000000059	TRUE	17576	17576	yes	1
MCDS_S_0000000060	TRUE	22360	22360	yes	1
MCDS_S_0000000061	TRUE	29025	29025	yes	1
MCDS_S_0000000062	TRUE	29945	29945	yes	1
MCDS_S_0000000063	TRUE	12741	12741	yes	1
MCDS_S_0000000064	TRUE	12833	12833	yes	1
MCDS_S_0000000065	TRUE	19917	19917	yes	1
MCDS_S_0000000066	TRUE	19641	19641	yes	1
MCDS_S_0000000067	TRUE	9015	9015	yes	1
MCDS_S_0000000068	TRUE	3817	3817	yes	1
MCDS_S_0000000069	TRUE	2897	2897	yes	1
MCDS_S_0000000070	TRUE	1379	1379	yes	1
MCDS_S_0000000071	TRUE	4277	4277	yes	1
MCDS_S_0000000072	TRUE	2529	2529	yes	1
MCDS_S_0000000073	TRUE	3587	3587	yes	1

MCDS_S_0000000074	TRUE	32613	32613	yes	1
MCDS_S_0000000075	TRUE	29163	29163	yes	1
MCDS_S_0000000076	TRUE	28432	28432	yes	1
MCDS_S_0000000077	TRUE	29260	29260	yes	1
MCDS_S_0000000078	TRUE	20474	20474	yes	1
MCDS_S_0000000079	TRUE	24154	24154	yes	1
MCDS_S_0000000080	TRUE	12562	12562	yes	1
MCDS_S_0000000081	TRUE	5386	5386	yes	1
MCDS_S_0000000082	TRUE	14448	14448	yes	1
MCDS_S_0000000083	TRUE	31146	31146	yes	1
MCDS_S_0000000084	TRUE	31376	31376	yes	1
MCDS_S_0000000085	TRUE	19687	19687	yes	1
MCDS_S_0000000086	TRUE	26868	26868	yes	1
MCDS_S_0000000087	TRUE	26960	26960	yes	1
MCDS_S_0000000088	TRUE	34550	34550	yes	1
MCDS_S_0000000089	TRUE	42692	42692	yes	1
MCDS_S_0000000090	TRUE	8284	8284	yes	1
MCDS_S_0000000091	TRUE	4696	4696	yes	1
MCDS_S_0000000092	TRUE	5892	5892	yes	1
MCDS_S_0000000093	TRUE	8652	8652	yes	1

MCDS_S_0000000094	TRUE	9526	9526	yes	1
MCDS_S_0000000095	TRUE	1154	1154	yes	1
MCDS_S_0000000096	TRUE	5708	5708	yes	1
MCDS_S_0000000097	TRUE	19600	19600	yes	1
MCDS_S_0000000098	TRUE	25212	25212	yes	1
MCDS_S_0000000099	TRUE	38644	38644	yes	1
MCDS_S_0000000100	TRUE	52168	52168	yes	1
MCDS_S_0000000101	TRUE	25442	25442	yes	1
MCDS_S_0000000102	TRUE	23740	23740	yes	1
MCDS_S_0000000103	TRUE	11642	11642	yes	1
MCDS_S_0000000104	TRUE	47200	47200	yes	1
MCDS_S_0000000105	TRUE	48074	48074	yes	1
MCDS_S_0000000106	TRUE	24982	24982	yes	1
MCDS_S_0000000107	TRUE	24568	24568	yes	1
MCDS_S_0000000108	TRUE	34044	34044	yes	1
MCDS_S_0000000109	TRUE	35746	35746	yes	1
MCDS_S_0000000110	TRUE	41818	41818	yes	1
MCDS_S_0000000111	TRUE	39099	39099	yes	1
MCDS_S_0000000112	TRUE	26679	26679	yes	1
MCDS_S_0000000113	TRUE	20515	20515	yes	1

MCDS_S_0000000114	TRUE	13247	13247	yes	1
MCDS_S_0000000115	TRUE	9383	9383	yes	1
MCDS_S_0000000116	TRUE	30727	30727	yes	1
MCDS_S_0000000117	TRUE	14402	14402	yes	1
MCDS_S_0000000118	TRUE	3868	3868	yes	1
MCDS_S_0000000119	TRUE	18726	18726	yes	1
MCDS_S_0000000120	TRUE	30226	30226	yes	1
MCDS_S_0000000121	TRUE	39881	39881	yes	1
MCDS_S_0000000122	TRUE	37765	37765	yes	1
MCDS_S_0000000123	TRUE	35327	35327	yes	1
MCDS_S_0000000124	TRUE	14264	14264	yes	1
MCDS_S_0000000125	TRUE	9756	9756	yes	1
MCDS_S_0000000126	TRUE	7962	7962	yes	1
MCDS_S_0000000127	TRUE	47563	47563	yes	1
MCDS_S_0000000128	TRUE	46873	46873	yes	1
MCDS_S_0000000129	TRUE	40111	40111	yes	1
MCDS_S_0000000130	TRUE	12971	12971	yes	1
MCDS_S_0000000131	TRUE	10211	10211	yes	1
MCDS_S_0000000132	TRUE	17249	17249	yes	1
MCDS_S_0000000133	TRUE	12833	12833	yes	1

MCDS_S_0000000134	TRUE	29623	29623	yes	1
MCDS_S_0000000135	TRUE	21849	21849	yes	1
MCDS_S_0000000136	TRUE	42503	42503	yes	1
MCDS_S_0000000137	TRUE	19692	19692	yes	1
MCDS_S_0000000138	TRUE	22314	22314	yes	1
MCDS_S_0000000139	TRUE	19094	19094	yes	1
MCDS_S_0000000140	TRUE	27006	27006	yes	1
MCDS_S_0000000141	TRUE	17852	17852	yes	1
MCDS_S_0000000142	TRUE	10216	10216	yes	1
MCDS_S_0000000143	TRUE	11596	11596	yes	1
MCDS_S_0000000144	TRUE	14770	14770	yes	1
MCDS_S_0000000145	TRUE	15092	15092	yes	1
MCDS_S_0000000146	TRUE	13160	13160	yes	1
MCDS_S_0000000147	TRUE	19646	19646	yes	1
MCDS_S_0000000148	TRUE	21440	21440	yes	1
MCDS_S_0000000149	TRUE	31238	31238	yes	1
MCDS_S_0000000150	TRUE	15782	15782	yes	1
MCDS_S_0000000151	TRUE	11458	11458	yes	1
MCDS_S_0000000152	TRUE	12240	12240	yes	1
MCDS_S_0000000153	TRUE	14034	14034	yes	1

MCDS_S_0000000154	TRUE	9158	9158	yes	1
MCDS_S_0000000155	TRUE	14586	14586	yes	1
MCDS_S_0000000156	TRUE	8008	8008	yes	1
MCDS_S_0000000157	TRUE	11550	11550	yes	1
MCDS_S_0000000158	TRUE	10722	10722	yes	1
MCDS_S_0000000159	TRUE	4006	4006	yes	1
MCDS_S_0000000160	TRUE	4972	4972	yes	1
MCDS_S_0000000161	TRUE	14264	14264	yes	1
MCDS_S_0000000162	TRUE	28202	28202	yes	1
MCDS_S_0000000163	TRUE	20520	20520	yes	1
MCDS_S_0000000164	TRUE	31928	31928	yes	1
MCDS_S_0000000165	TRUE	9756	9756	yes	1
MCDS_S_0000000166	TRUE	5524	5524	yes	1
MCDS_S_0000000167	TRUE	1982	1982	yes	1
MCDS_S_0000000168	TRUE	27001	27001	yes	1
MCDS_S_0000000169	TRUE	30451	30451	yes	1
MCDS_S_0000000170	TRUE	19738	19738	yes	1
MCDS_S_0000000171	TRUE	3730	3730	yes	1
MCDS_S_0000000172	TRUE	22866	22866	yes	1
MCDS_S_0000000173	TRUE	15690	15690	yes	1

MCDS_S_0000000174	TRUE	29306	29306	yes	1
MCDS_S_0000000175	TRUE	20566	20566	yes	1
MCDS_S_0000000176	TRUE	23372	23372	yes	1
MCDS_S_0000000177	TRUE	13942	13942	yes	1
MCDS_S_0000000178	TRUE	23694	23694	yes	1
MCDS_S_0000000179	TRUE	14356	14356	yes	1
MCDS_S_0000000180	TRUE	22268	22268	yes	1
MCDS_S_0000000181	TRUE	18404	18404	yes	1
MCDS_S_0000000182	TRUE	15920	15920	yes	1
MCDS_S_0000000183	TRUE	23372	23372	yes	1
MCDS_S_0000000184	TRUE	15598	15598	yes	1
MCDS_S_0000000185	TRUE	27052	27052	yes	1
MCDS_S_0000000186	TRUE	28432	28432	yes	1
MCDS_S_0000000187	TRUE	22774	22774	yes	1
MCDS_S_0000000188	TRUE	23464	23464	yes	1
MCDS_S_0000000189	TRUE	16334	16334	yes	1
MCDS_S_0000000190	TRUE	17162	17162	yes	1
MCDS_S_0000000191	TRUE	12884	12884	yes	1
MCDS_S_0000000192	TRUE	11780	11780	yes	1
MCDS_S_0000000193	TRUE	5800	5800	yes	1

MCDS_S_0000000194	TRUE	1890	1890	yes	1
MCDS_S_0000000195	TRUE	1614	1614	yes	1
MCDS_S_0000000196	TRUE	8284	8284	yes	1
MCDS_S_0000000197	TRUE	12102	12102	yes	1
MCDS_S_0000000198	TRUE	9618	9618	yes	1
MCDS_S_0000000199	TRUE	4282	4282	yes	1
MCDS_S_0000000200	TRUE	19416	19416	yes	1
MCDS_S_0000000201	TRUE	12884	12884	yes	1
MCDS_S_0000000202	TRUE	13942	13942	yes	1
MCDS_S_0000000203	TRUE	15966	15966	yes	1
MCDS_S_0000000204	TRUE	37448	37448	yes	1
MCDS_S_0000000205	TRUE	38782	38782	yes	1
MCDS_S_0000000206	TRUE	23280	23280	yes	1
MCDS_S_0000000207	TRUE	21946	21946	yes	1
MCDS_S_0000000208	TRUE	15092	15092	yes	1
MCDS_S_0000000209	TRUE	11136	11136	yes	1
MCDS_S_0000000210	TRUE	42784	42784	yes	1
MCDS_S_0000000211	TRUE	34228	34228	yes	1
MCDS_S_0000000212	TRUE	12378	12378	yes	1
MCDS_S_0000000213	TRUE	15782	15782	yes	1

MCDS_S_0000000214	TRUE	21619	21619	yes	1
MCDS_S_0000000215	TRUE	7548	7548	yes	1
MCDS_S_0000000216	TRUE	3546	3546	yes	1
MCDS_S_0000000217	TRUE	1752	1752	yes	1
MCDS_S_0000000218	TRUE	5248	5248	yes	1
MCDS_S_0000000219	TRUE	5519	5519	yes	1
MCDS_S_0000000220	TRUE	9567	9567	yes	1
MCDS_S_0000000221	TRUE	14857	14857	yes	1
MCDS_S_0000000222	TRUE	15690	15690	yes	1
MCDS_S_0000000223	TRUE	14448	14448	yes	1
MCDS_S_0000000224	TRUE	26224	26224	yes	1
MCDS_S_0000000225	TRUE	21394	21394	yes	1
MCDS_S_0000000226	TRUE	22820	22820	yes	1
MCDS_S_0000000227	TRUE	37724	37724	yes	1
MCDS_S_0000000228	TRUE	13482	13482	yes	1
MCDS_S_0000000229	TRUE	30088	30088	yes	1
MCDS_S_0000000230	TRUE	20520	20520	yes	1
MCDS_S_0000000231	TRUE	42324	42324	yes	1
MCDS_S_0000000232	TRUE	21302	21302	yes	1
MCDS_S_0000000233	TRUE	33308	33308	yes	1

MCDS_S_0000000234	TRUE	36482	36482	yes	1
MCDS_S_0000000235	TRUE	32802	32802	yes	1
MCDS_S_0000000236	TRUE	22452	22452	yes	1
MCDS_S_0000000237	TRUE	25856	25856	yes	1
MCDS_S_0000000238	TRUE	29674	29674	yes	1
MCDS_S_0000000239	TRUE	4691	4691	yes	1
MCDS_S_0000000240	TRUE	1747	1747	yes	1
MCDS_S_0000000241	TRUE	5473	5473	yes	1
MCDS_S_0000000242	TRUE	1839	1839	yes	1
MCDS_S_0000000243	TRUE	10947	10947	yes	1
MCDS_S_0000000244	TRUE	20014	20014	yes	1
MCDS_S_0000000245	TRUE	25442	25442	yes	1
MCDS_S_0000000246	TRUE	23418	23418	yes	1
MCDS_S_0000000247	TRUE	18450	18450	yes	1
MCDS_S_0000000248	TRUE	42554	42554	yes	1
MCDS_S_0000000249	TRUE	29766	29766	yes	1
MCDS_S_0000000250	TRUE	19089	19089	yes	1
MCDS_S_0000000251	TRUE	33763	33763	yes	1
MCDS_S_0000000252	TRUE	28151	28151	yes	1
MCDS_S_0000000253	TRUE	30083	30083	yes	1

MCDS_S_0000000254	TRUE	47517	47517	yes	1
MCDS_S_0000000255	TRUE	47931	47931	yes	1
MCDS_S_0000000256	TRUE	16978	16978	yes	1
MCDS_S_0000000257	TRUE	12976	12976	yes	1
MCDS_S_0000000258	TRUE	5892	5892	yes	1
MCDS_S_0000000259	TRUE	18082	18082	yes	1
MCDS_S_0000000260	TRUE	15184	15184	yes	1
MCDS_S_0000000261	TRUE	17024	17024	yes	1
MCDS_S_0000000262	TRUE	23924	23924	yes	1
MCDS_S_0000000263	TRUE	12102	12102	yes	1
MCDS_S_0000000264	TRUE	49362	49362	yes	1
MCDS_S_0000000265	TRUE	32020	32020	yes	1
MCDS_S_0000000266	TRUE	42462	42462	yes	1
MCDS_S_0000000267	TRUE	45544	45544	yes	1
MCDS_S_0000000268	TRUE	10722	10722	yes	1
MCDS_S_0000000269	TRUE	16748	16748	yes	1
MCDS_S_0000000270	TRUE	14770	14770	yes	1
MCDS_S_0000000271	TRUE	21118	21118	yes	1
MCDS_S_0000000272	TRUE	46689	46689	yes	1
MCDS_S_0000000273	TRUE	28473	28473	yes	1

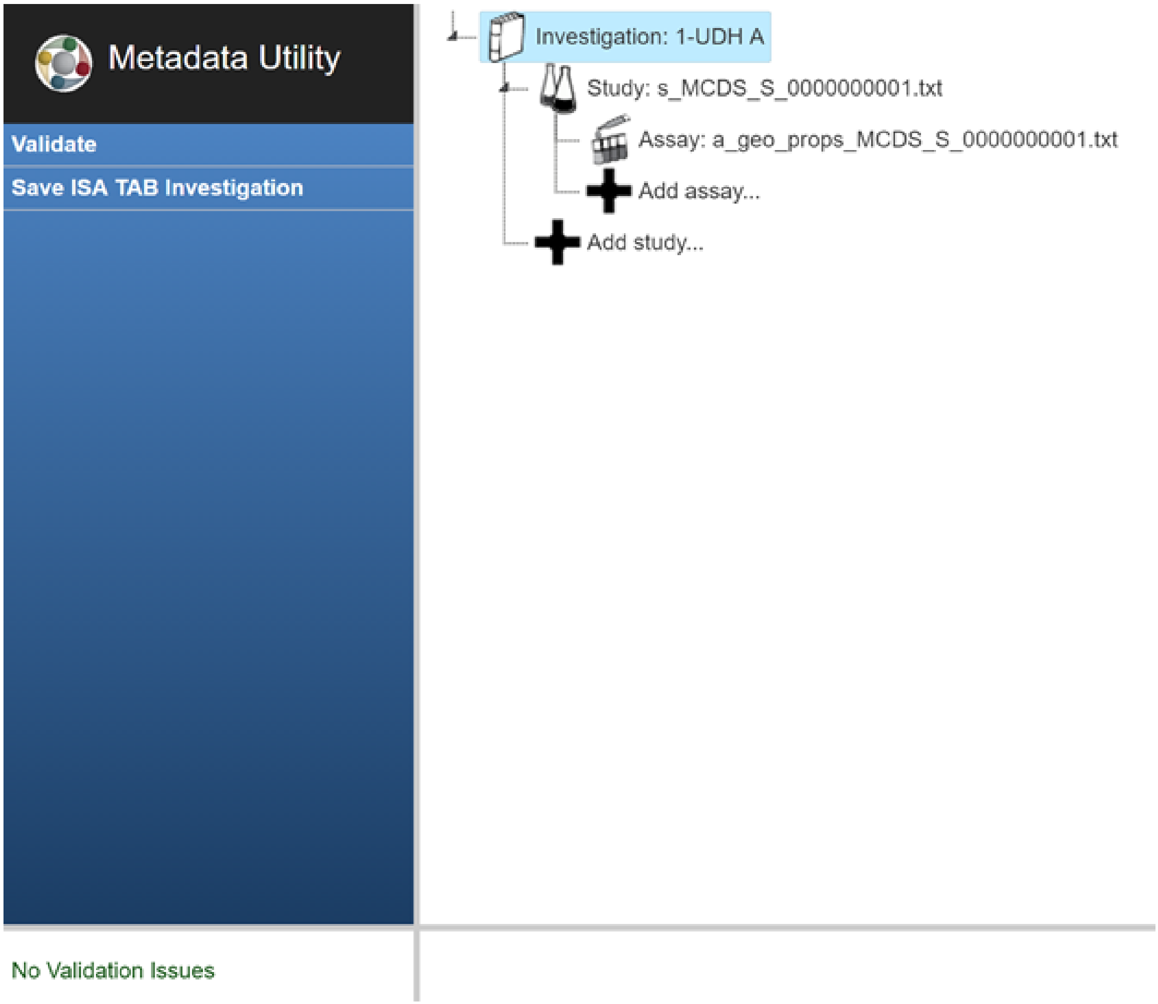
MCDS_S_0000000274	TRUE	18445	18445	yes	1
MCDS_S_0000000275	TRUE	15225	15225	yes	1
MCDS_S_0000000276	TRUE	23183	23183	yes	1
MCDS_S_0000000277	TRUE	21067	21067	yes	1
MCDS_S_0000000278	TRUE	13017	13017	yes	1
MCDS_S_0000000279	TRUE	30727	30727	yes	1
MCDS_S_0000000280	TRUE	18077	18077	yes	1
MCDS_S_0000000281	TRUE	10027	10027	yes	1
MCDS_S_0000000282	TRUE	43377	43377	yes	1
MCDS_S_0000000283	TRUE	36339	36339	yes	1
MCDS_S_0000000284	TRUE	29255	29255	yes	1
MCDS_S_0000000285	TRUE	42733	42733	yes	1
MCDS_S_0000000286	TRUE	28197	28197	yes	1
MCDS_S_0000000287	TRUE	34315	34315	yes	1
MCDS_S_0000000288	TRUE	21067	21067	yes	1
MCDS_S_0000000289	TRUE	39007	39007	yes	1
MCDS_S_0000000290	TRUE	27461	27461	yes	1
MCDS_S_0000000291	TRUE	35741	35741	yes	1
MCDS_S_0000000292	TRUE	11131	11131	yes	1
MCDS_S_0000000293	TRUE	17111	17111	yes	1

MCDS_S_0000000294	TRUE	10809	10809	yes	1
MCDS_S_0000000295	TRUE	15731	15731	yes	1
MCDS_S_0000000296	TRUE	14259	14259	yes	1
MCDS_S_0000000297	TRUE	25299	25299	yes	1
MCDS_S_0000000298	TRUE	25483	25483	yes	1
MCDS_S_0000000299	TRUE	23321	23321	yes	1
MCDS_S_0000000300	TRUE	17847	17847	yes	1
MCDS_S_0000000301	TRUE	24471	24471	yes	1
MCDS_S_0000000302	TRUE	23275	23275	yes	1
MCDS_S_0000000303	TRUE	27047	27047	yes	1
MCDS_S_0000000304	TRUE	18077	18077	yes	1
MCDS_S_0000000305	TRUE	20193	20193	yes	1
MCDS_S_0000000306	TRUE	22263	22263	yes	1
MCDS_S_0000000307	TRUE	22079	22079	yes	1
MCDS_S_0000000308	TRUE	20009	20009	yes	1
MCDS_S_0000000309	TRUE	21435	21435	yes	1
MCDS_S_0000000310	TRUE	23505	23505	yes	1
MCDS_S_0000000311	TRUE	25759	25759	yes	1
MCDS_S_0000000312	TRUE	25943	25943	yes	1
MCDS_S_0000000313	TRUE	28427	28427	yes	1

MCDS_S_0000000314	TRUE	25023	25023	yes	1
MCDS_S_0000000315	TRUE	27967	27967	yes	1
MCDS_S_0000000316	TRUE	24655	24655	yes	1
MCDS_S_0000000317	TRUE	25943	25943	yes	1
MCDS_S_0000000318	TRUE	15777	15777	yes	1
MCDS_S_0000000319	TRUE	18997	18997	yes	1
MCDS_S_0000000320	TRUE	24103	24103	yes	1
MCDS_S_0000000321	TRUE	19871	19871	yes	1
MCDS_S_0000000322	TRUE	19273	19273	yes	1
MCDS_S_0000000323	TRUE	21895	21895	yes	1
MCDS_S_0000000324	TRUE	30589	30589	yes	1
MCDS_S_0000000325	TRUE	30543	30543	yes	1
MCDS_S_0000000326	TRUE	18261	18261	yes	1
MCDS_S_0000000327	TRUE	29669	29669	yes	1

APPENDIX III.

APPENDIX III contains screenshots of proof of validation of each ISA-Tab file set selected for validation. This selection of ISA-Tab file sets constitutes 10% of all ISA-Tab file sets produced, and represents all entities mapped between MCDS DSSs and ISA-Tab Format

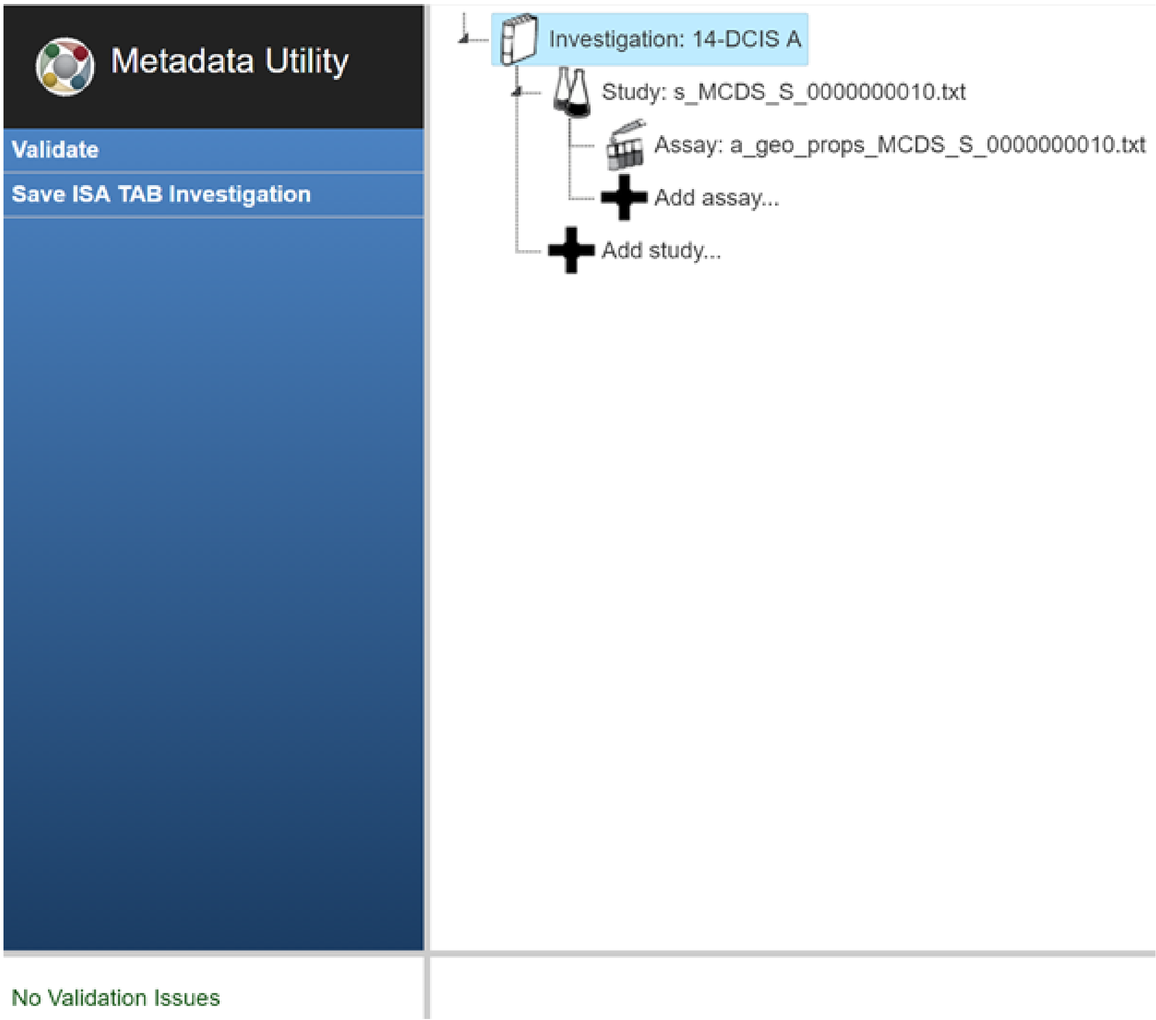


Investigation: 1-UDH A

Study: s_MCDS_S_0000000001.txt

Assay: a_geo_props_MCDS_S_0000000001.txt

No Validation Issues

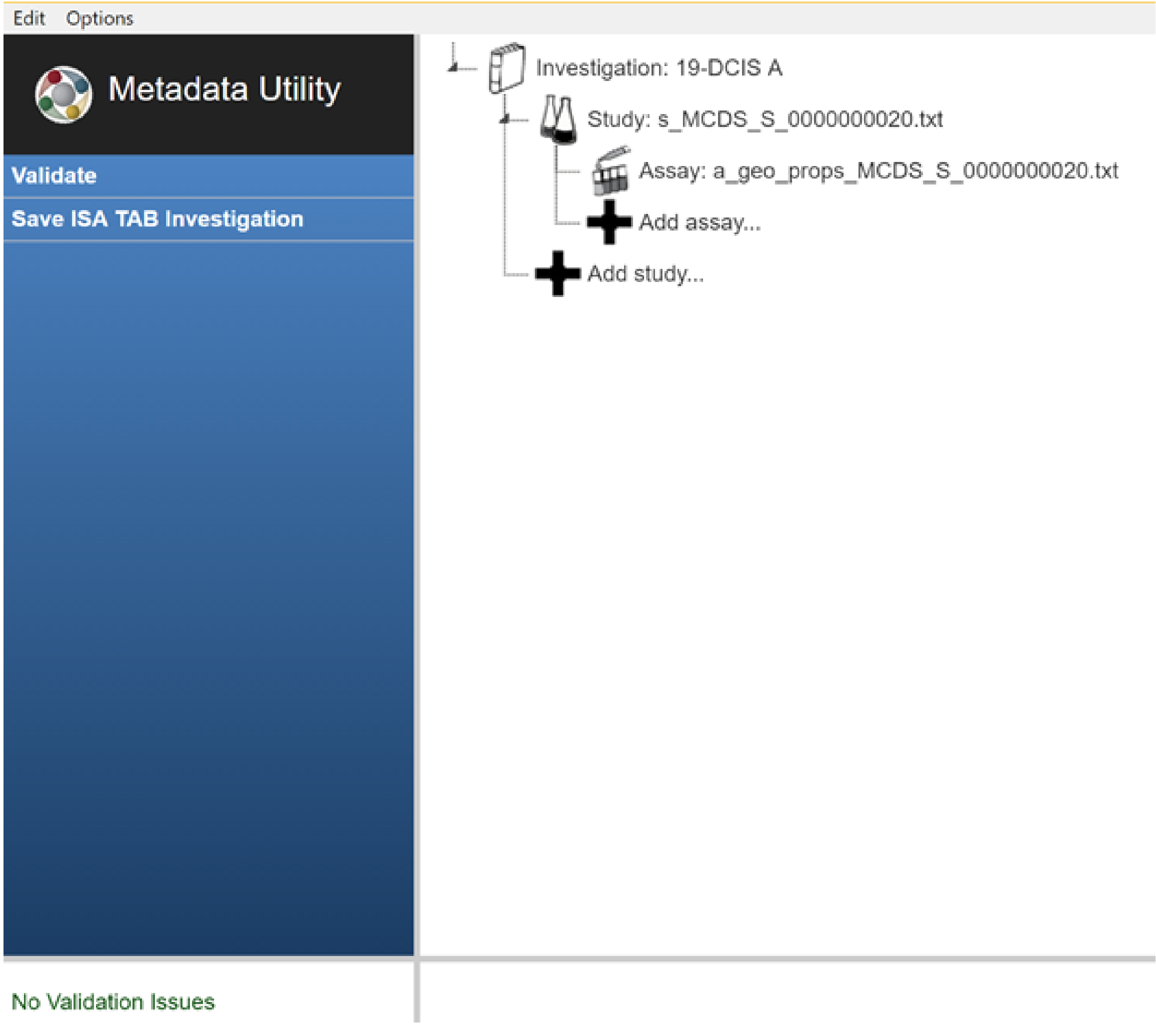


Investigation: 14-DCIS A

Study: s_MCDS_S_0000000010.txt

Assay: a_geo_props_MCDS_S_0000000010.txt

No Validation Issues

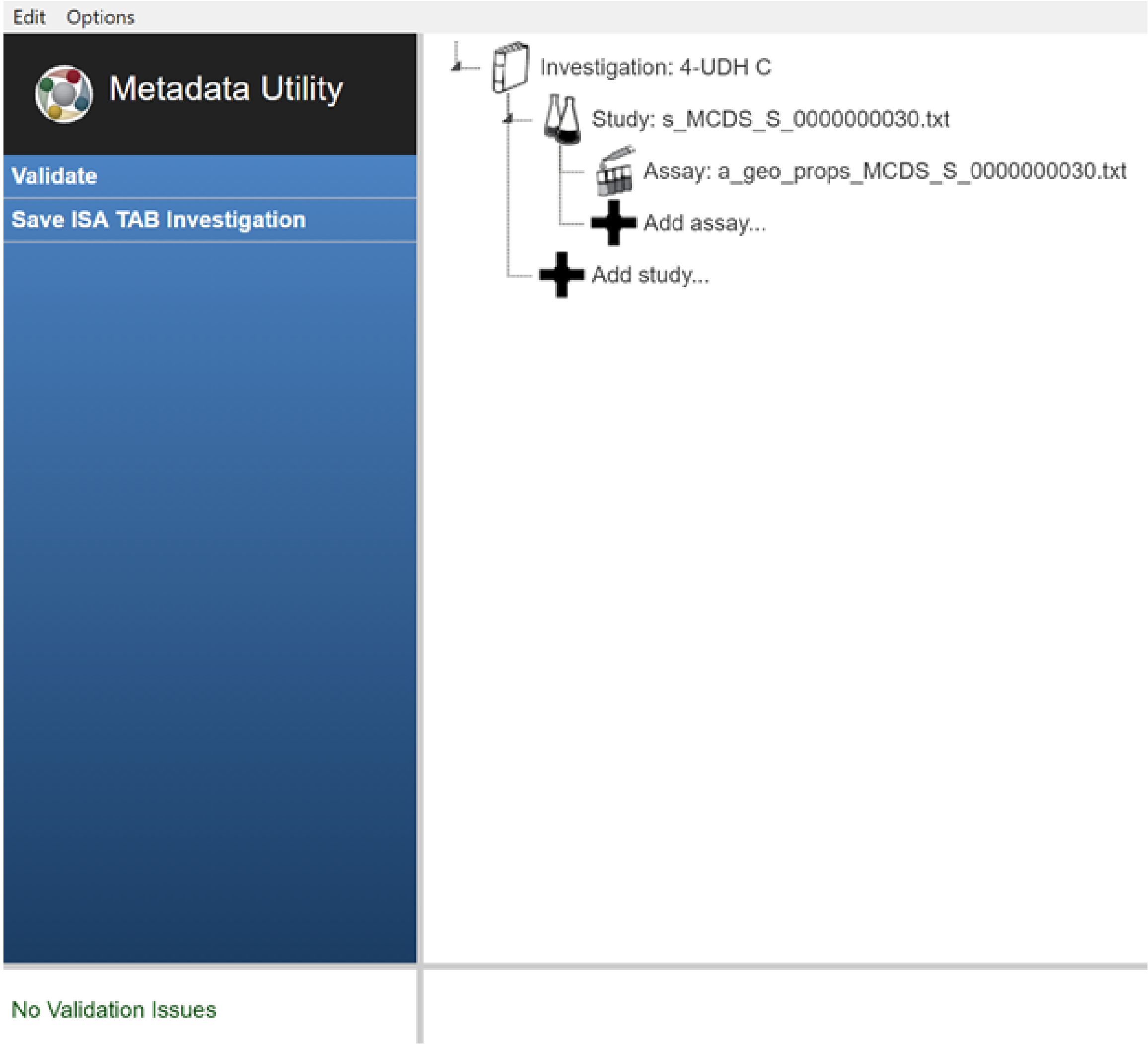


Investigation: 19-DCIS A

Study: s_MCDS_S_0000000020.txt

Assay: a_geo_props_MCDS_S_0000000020.txt

No Validation Issues

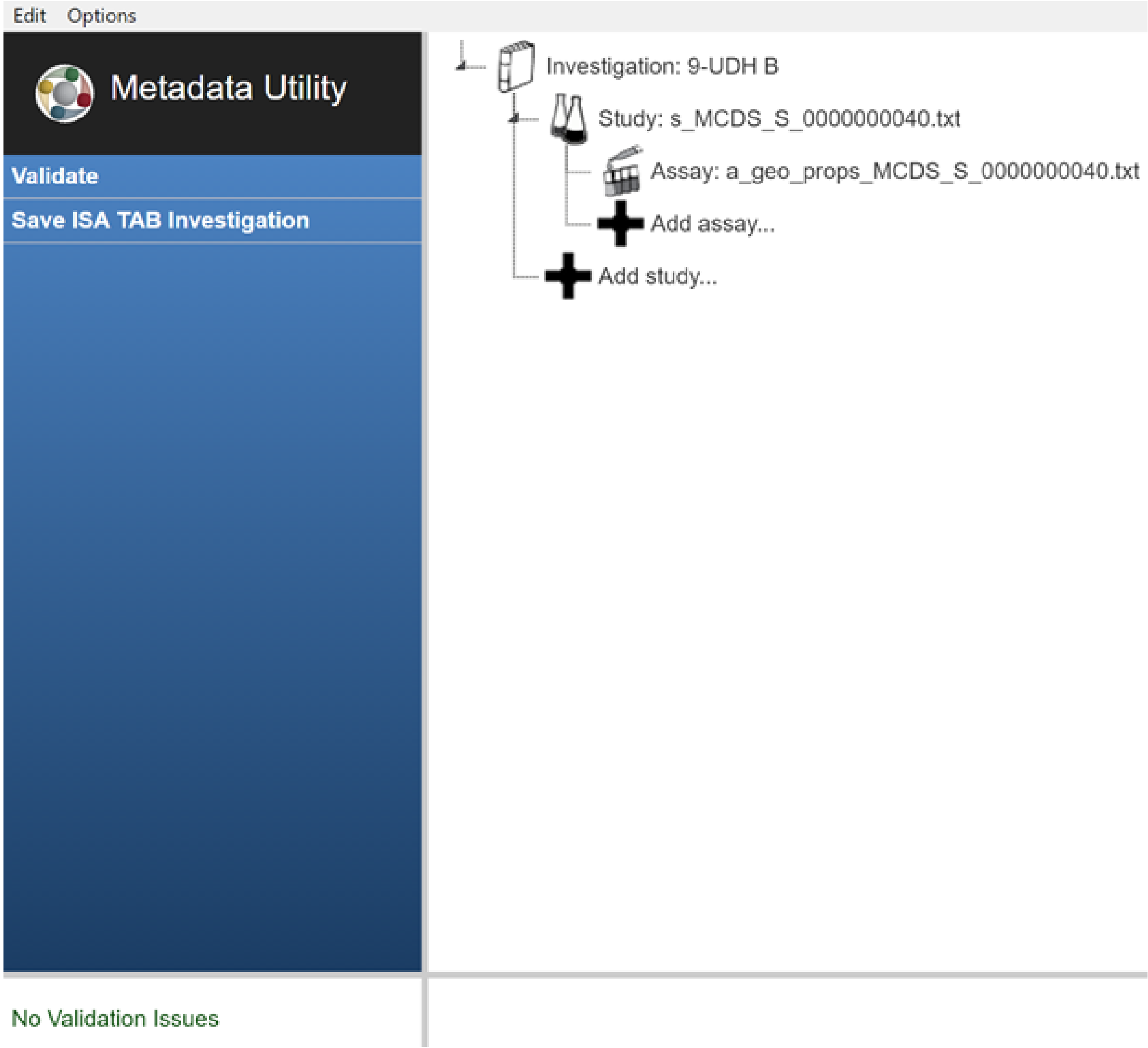


Investigation: 4-UDH C

Study: s_MCDS_S_0000000030.txt

Assay: a_geo_props_MCDS_S_0000000030.txt

No Validation Issues

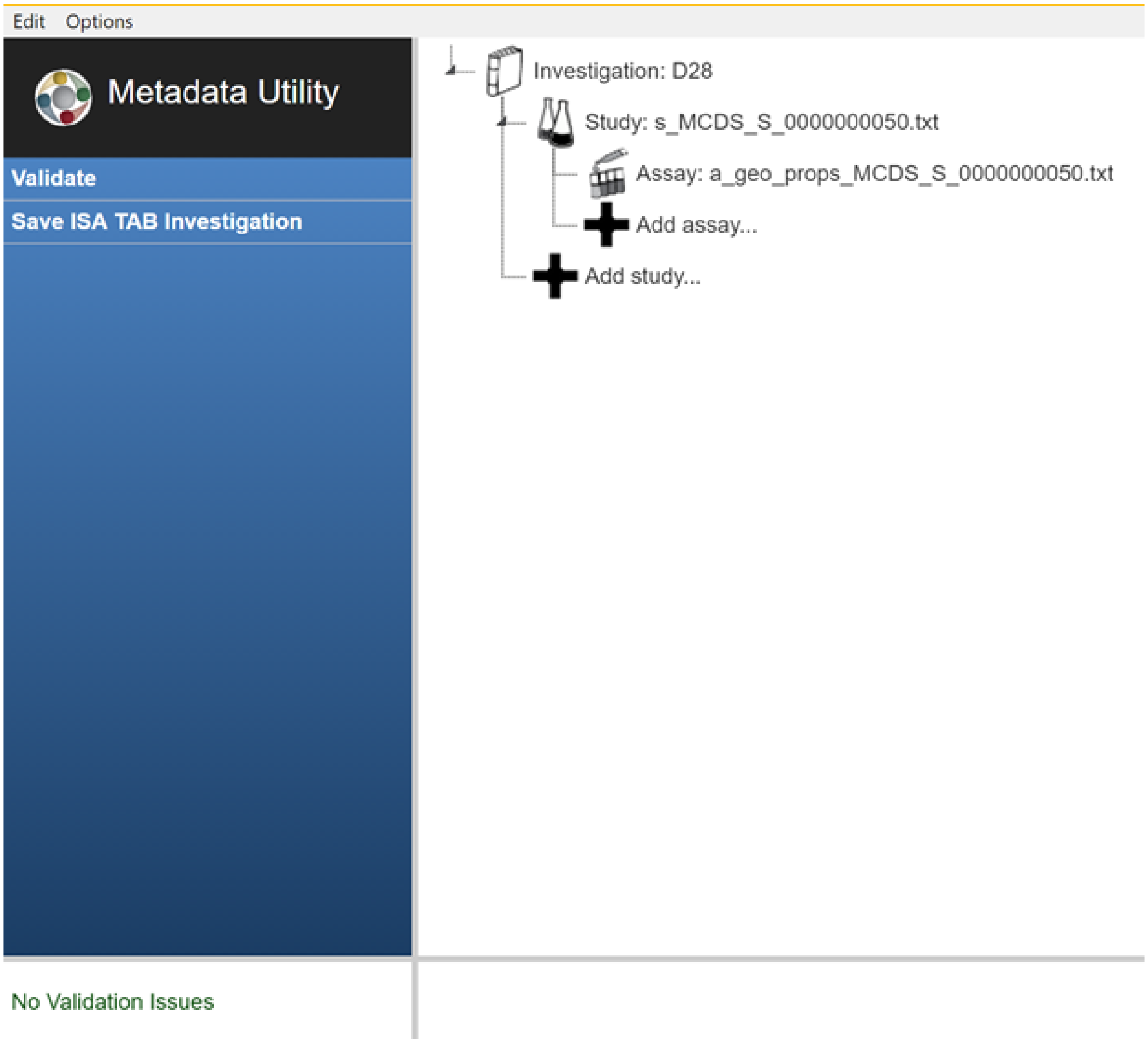


Investigation: 9-UDH B

Study: s_MCDS_S_0000000040.txt

Assay: a_geo_props_MCDS_S_0000000040.txt

No Validation Issues

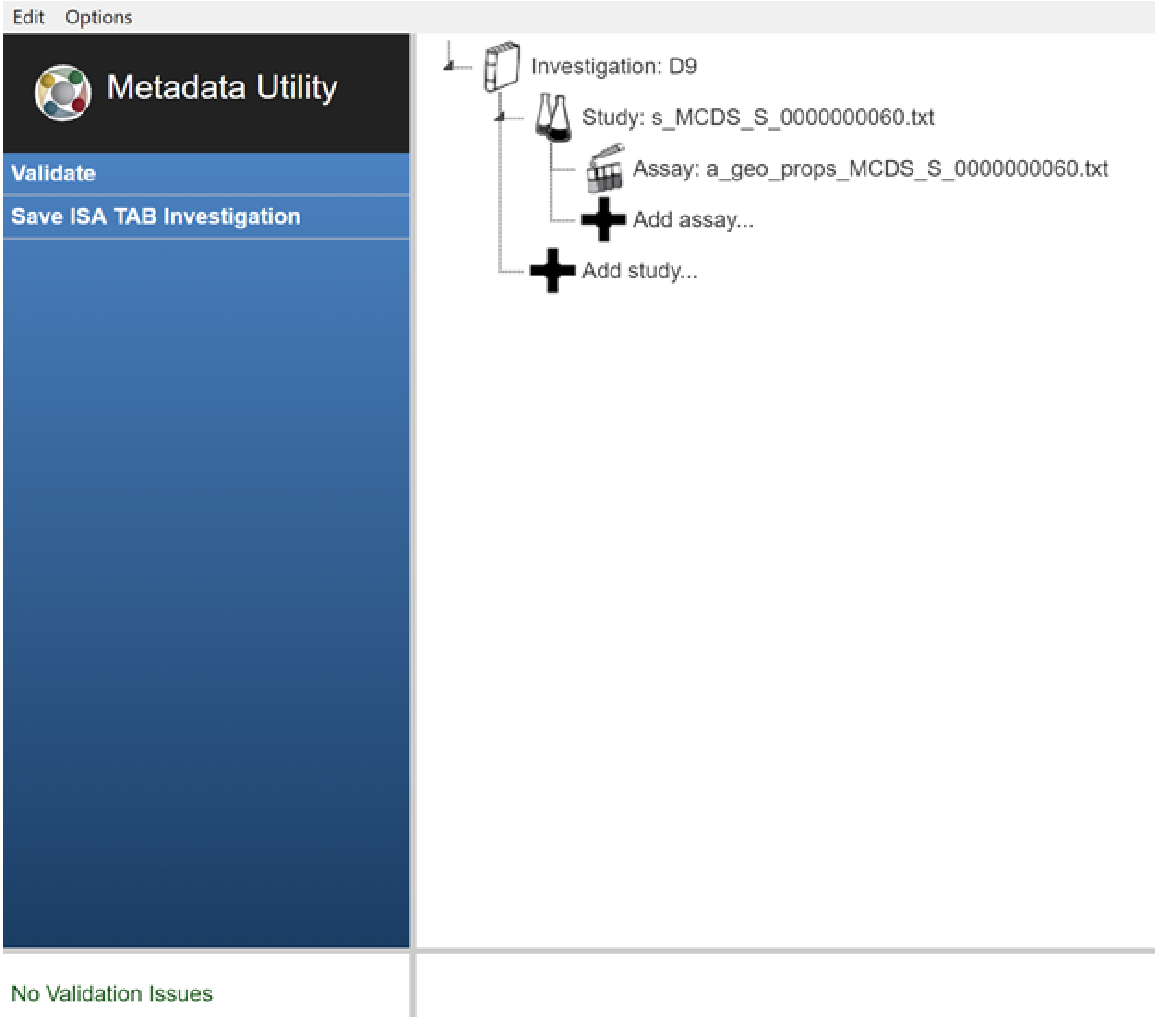


Investigation: D28

Study: s_MCDS_S_0000000050.txt

Assay: a_geo_props_MCDS_S_0000000050.txt

No Validation Issues



Investigation: D9

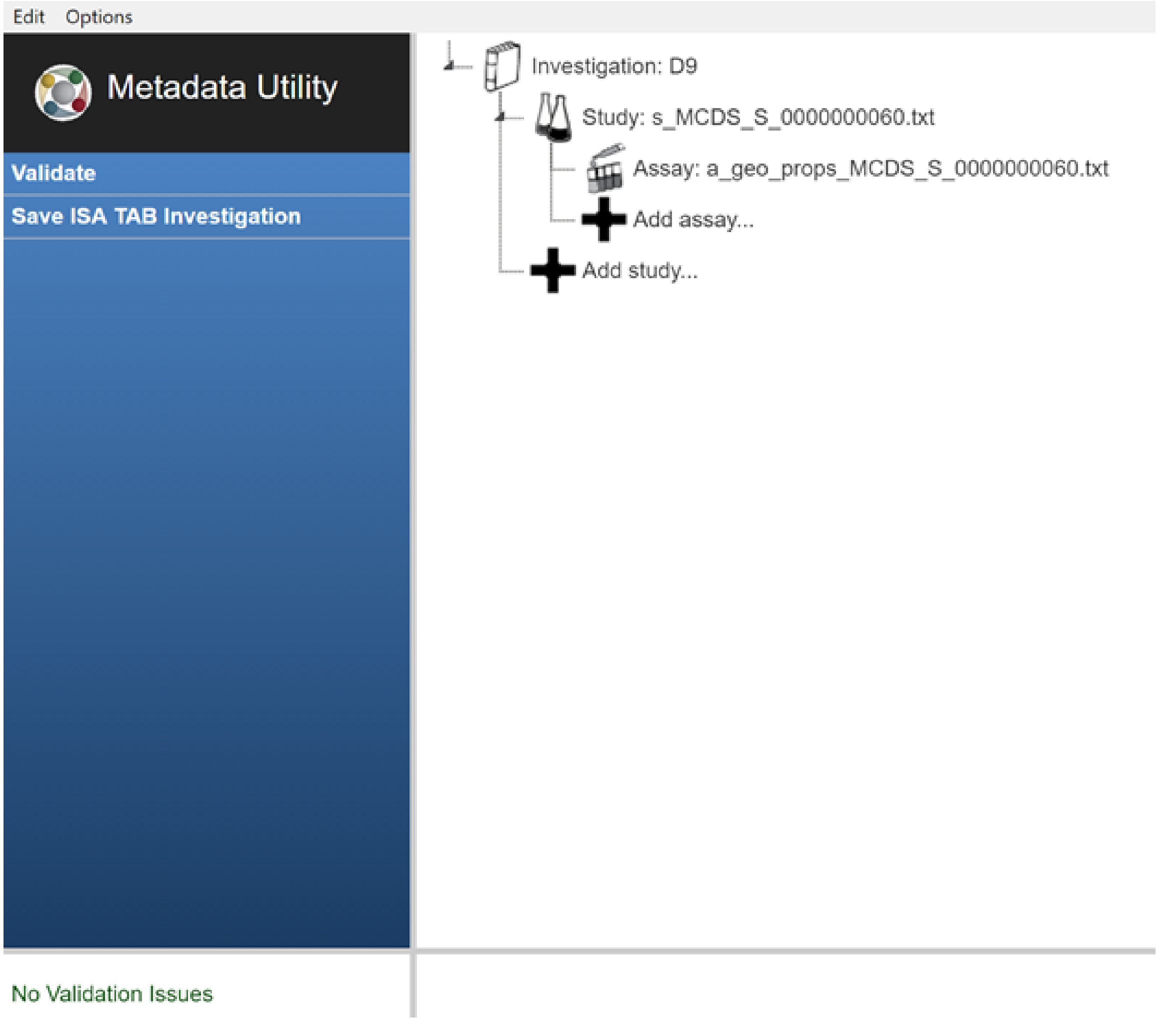
Study: s_MCDS_S_0000000060.txt

Assay: a_geo_props_MCDS_S_0000000060.txt

Add assay...

Add study...

No Validation Issues

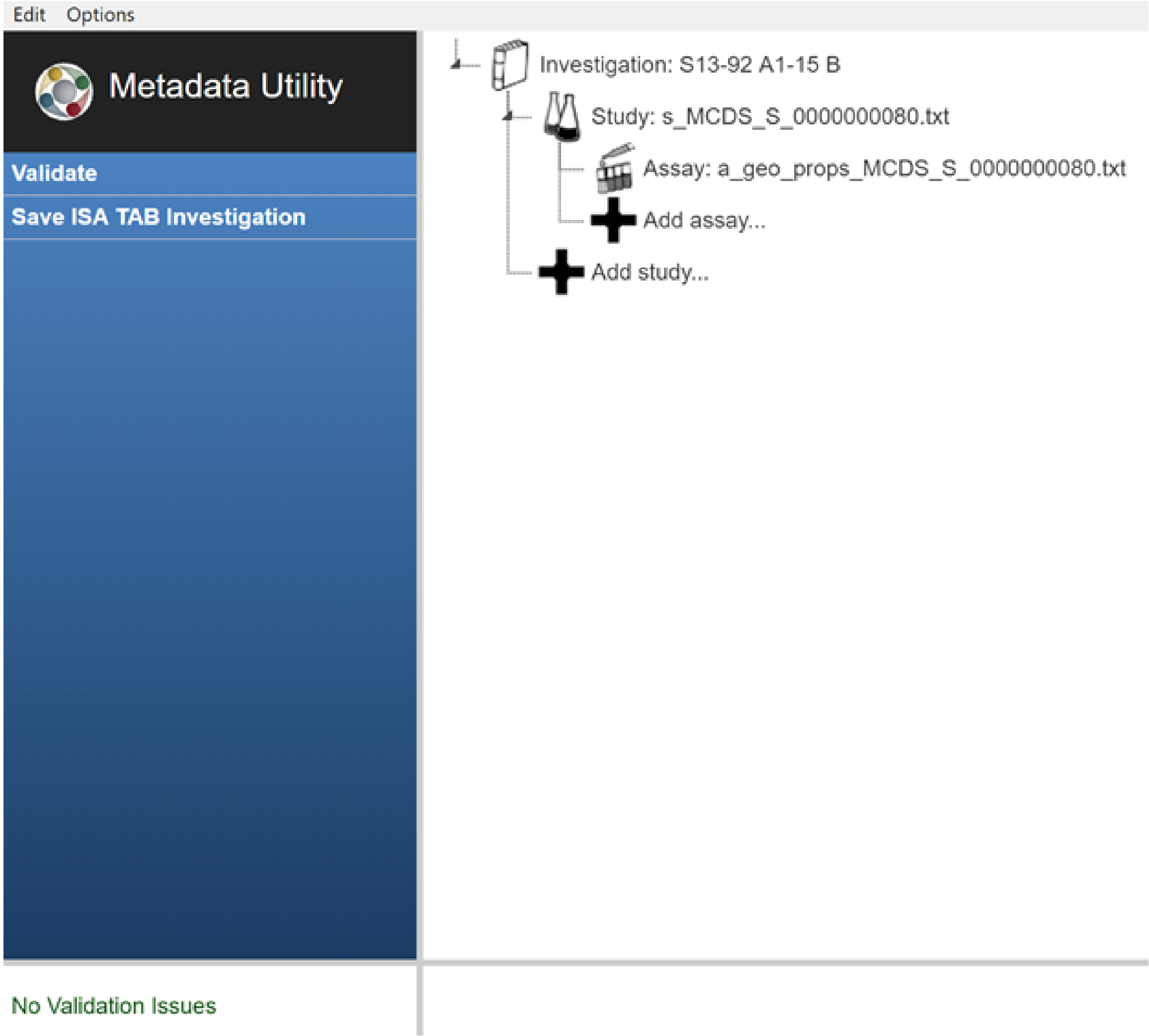


Investigation: D9

Study: s_MCDS_S_0000000060.txt

Assay: a_geo_props_MCDS_S_0000000060.txt

No Validation Issues

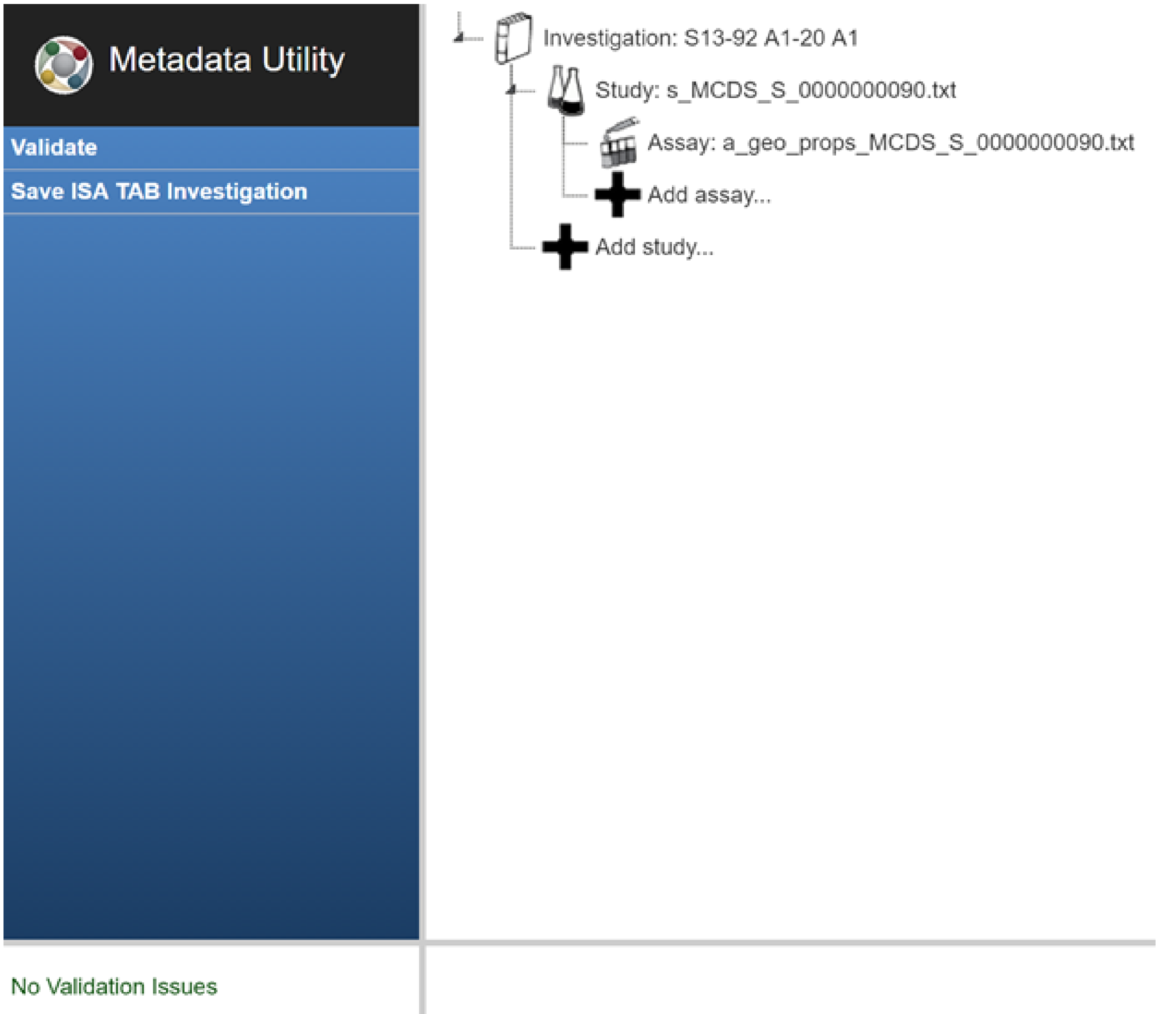


Investigation: S13-92 A1-15 B

Study: s_MCDS_S_0000000080.txt

Assay: a_geo_props_MCDS_S_0000000080.txt

No Validation Issues

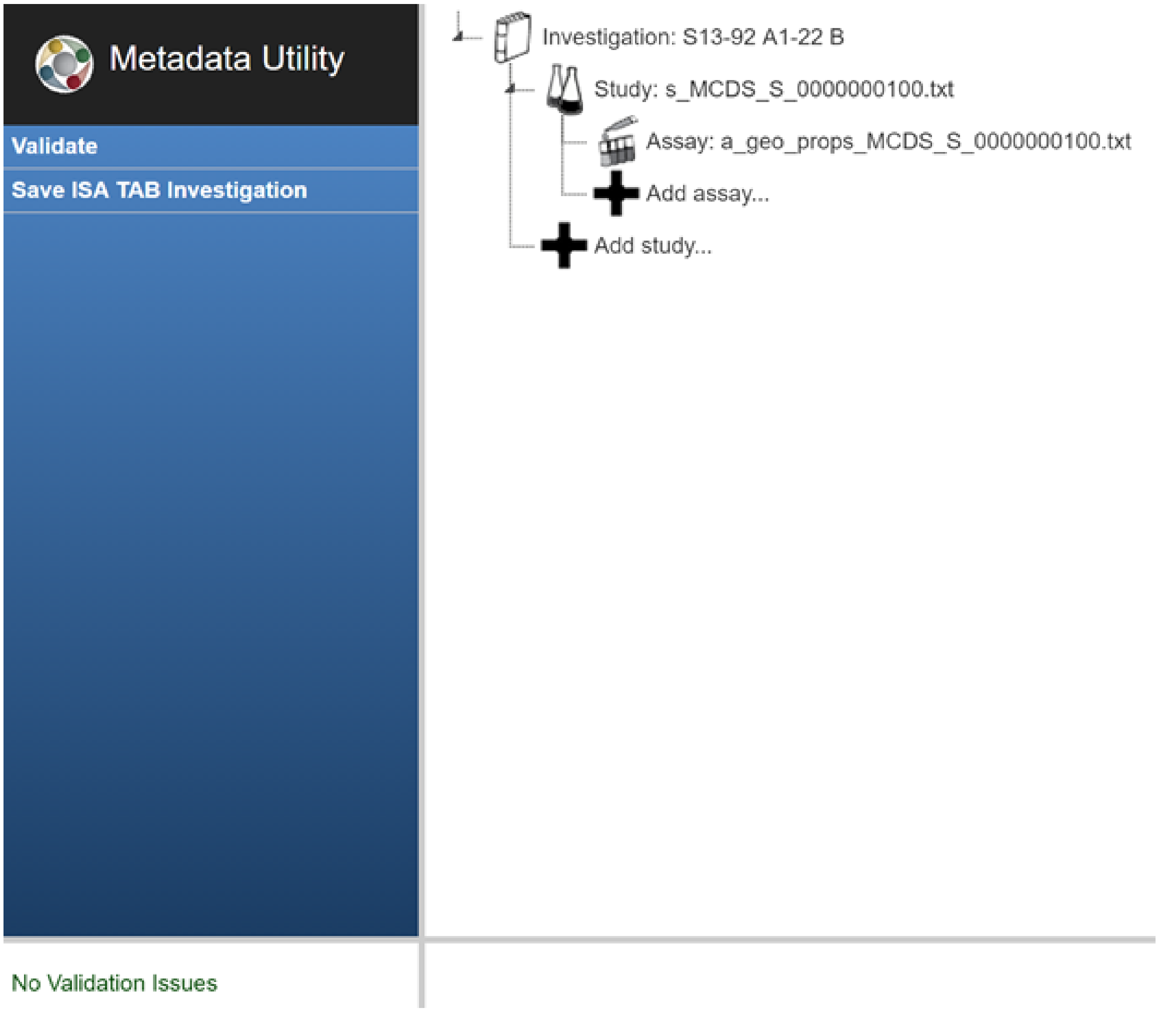


Investigation: S13-92 A1-20 A1

Study: s_MCDS_S_0000000090.txt

Assay: a_geo_props_MCDS_S_0000000090.txt

No Validation Issues

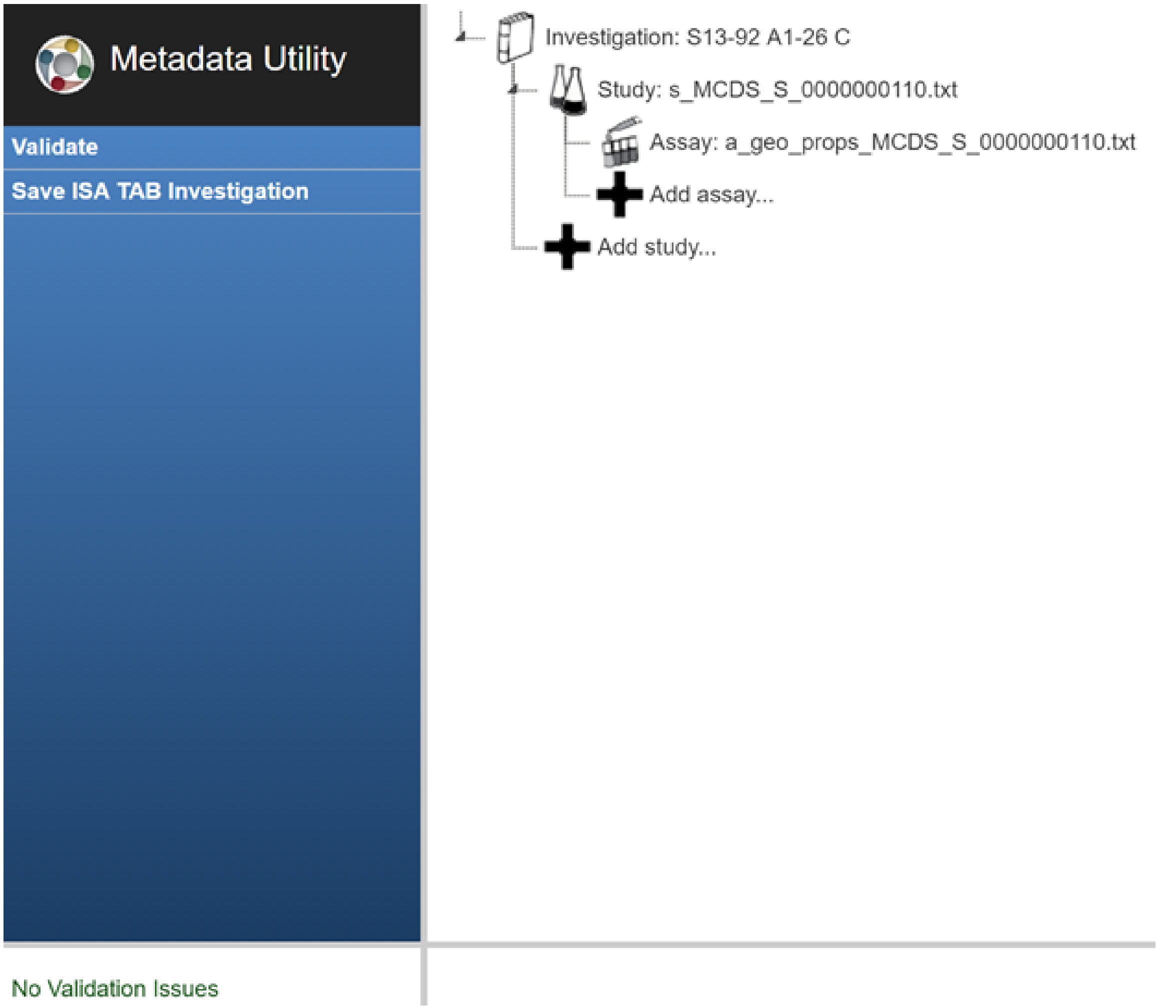


Investigation: S13-92 A1-22 B

Study: s_MCDS_S_0000000100.txt

Assay: a_geo_props_MCDS_S_0000000100.txt

No Validation Issues

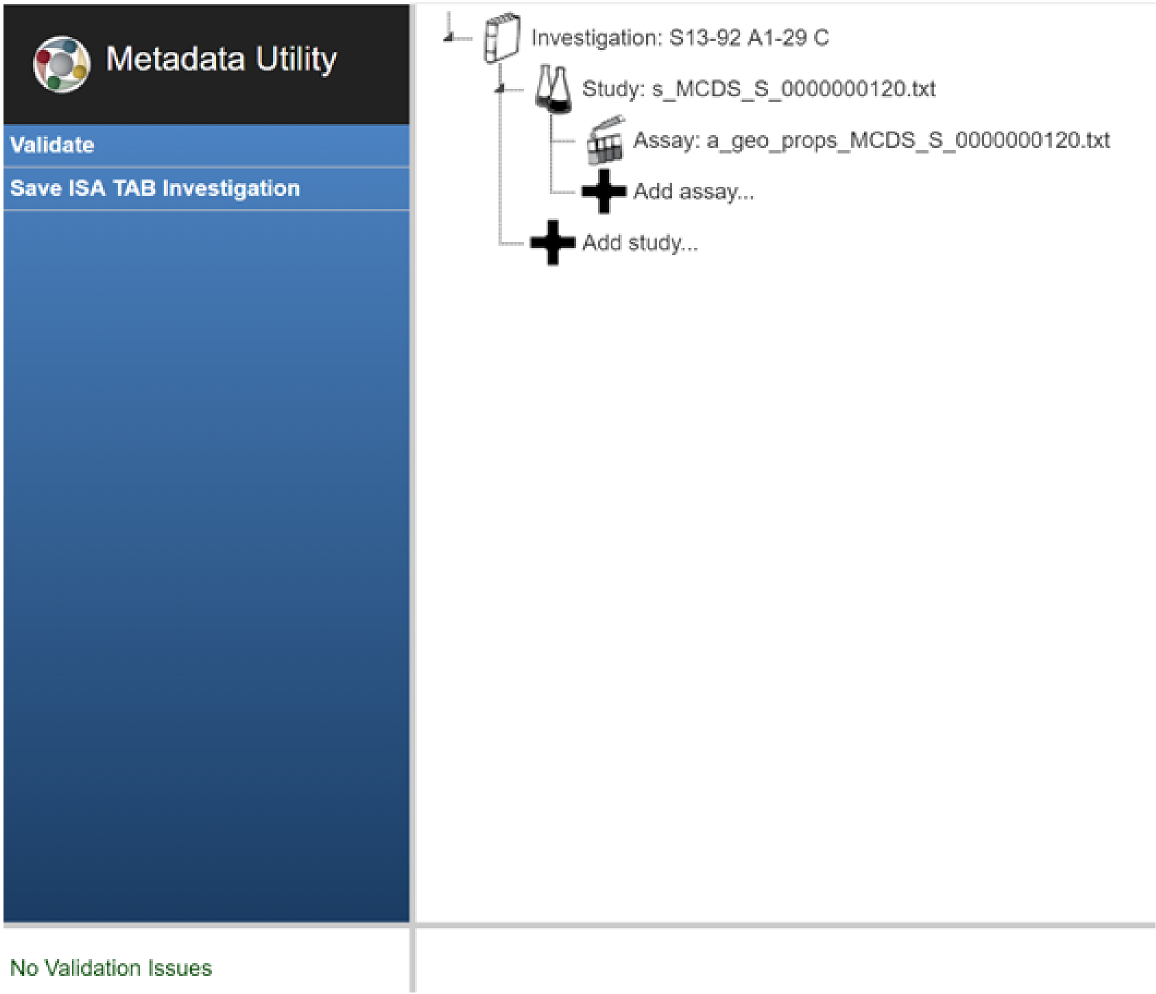


Investigation: S13-92 A1-26 C

Study: s_MCDS_S_0000000110.txt

Assay: a_geo_props_MCDS_S_0000000110.txt

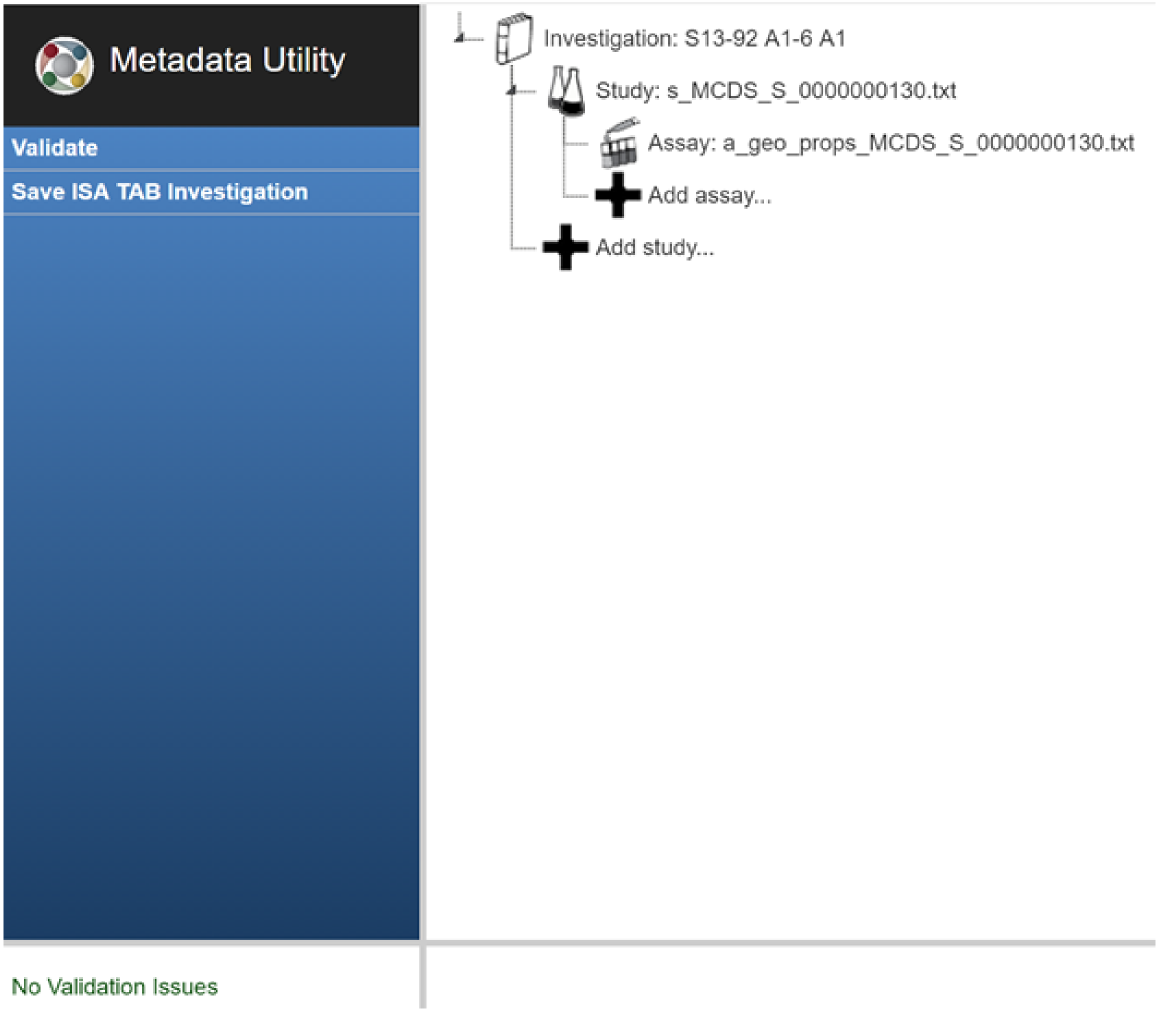
No Validation Issues



Investigation: S13-92 A1-29 C

Study: s_MCDS_S_0000000120.txt

Assay: a_geo_props_MCDS_S_0000000120.txt



Investigation: S13-92 A1-6 A1

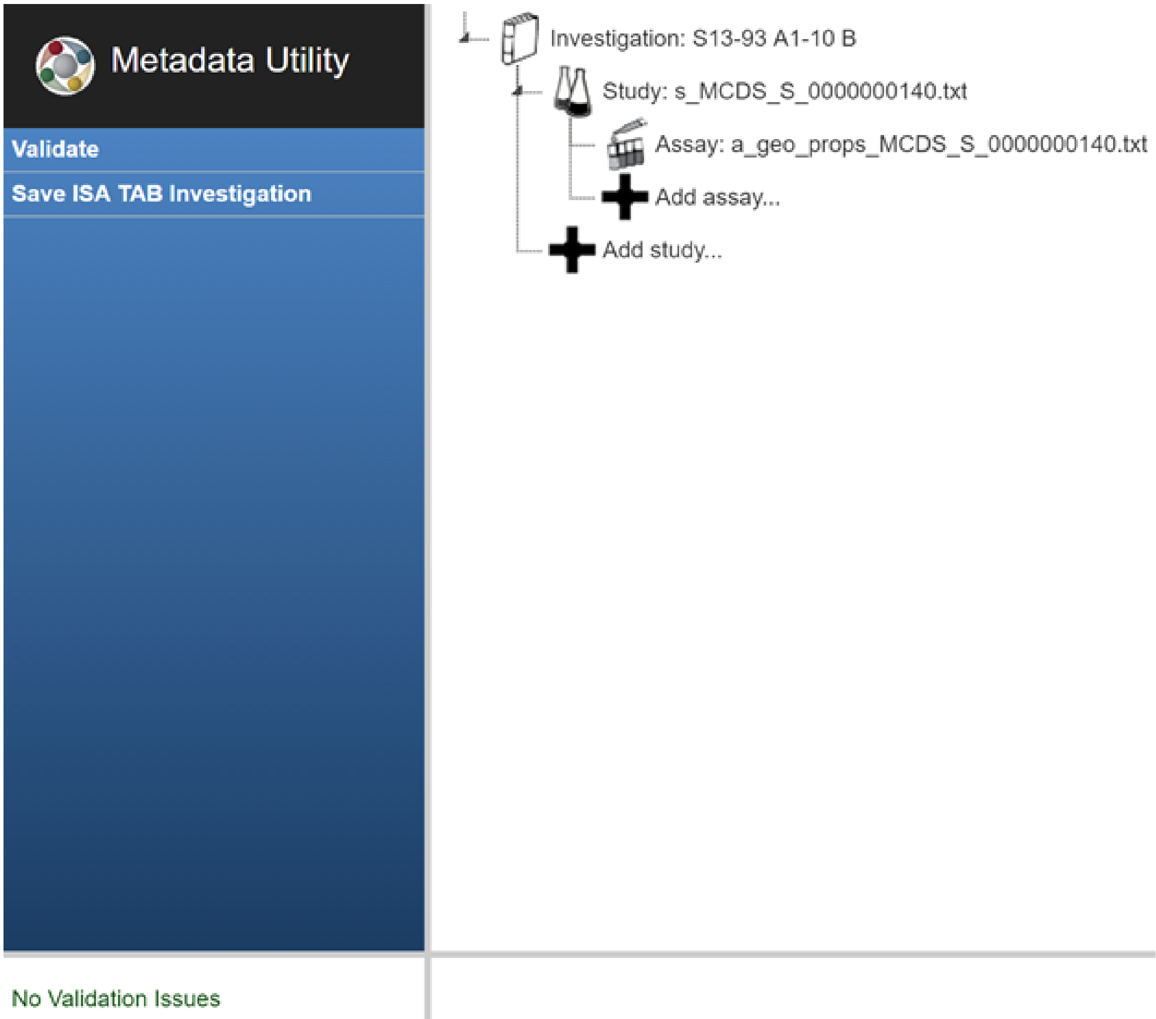
Study: s_MCDS_S_0000000130.txt

Assay: a_geo_props_MCDS_S_0000000130.txt

Add assay...

Add study...

No Validation Issues

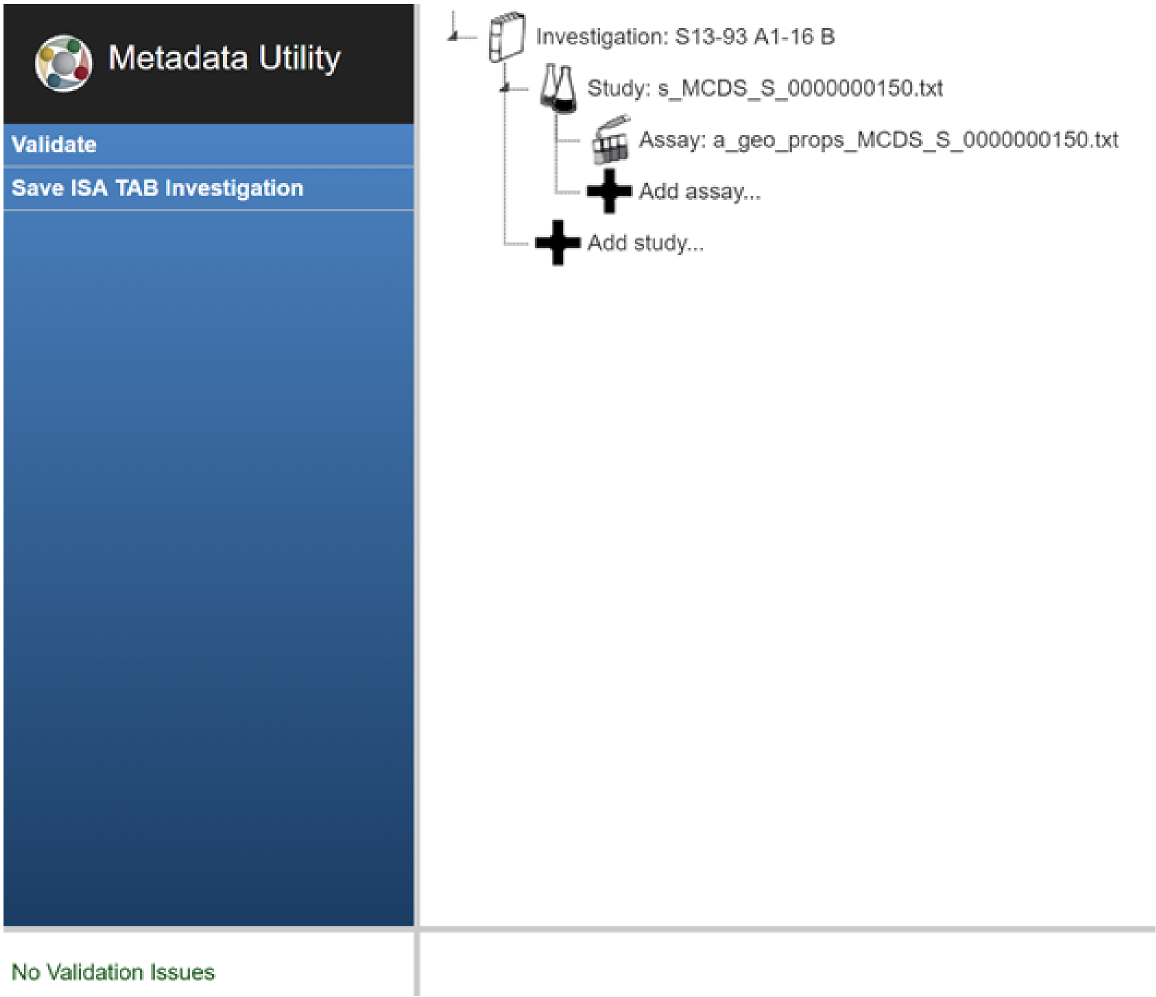


Investigation: S13-93 A1-10 B

Study: s_MCDS_S_0000000140.txt

Assay: a_geo_props_MCDS_S_0000000140.txt

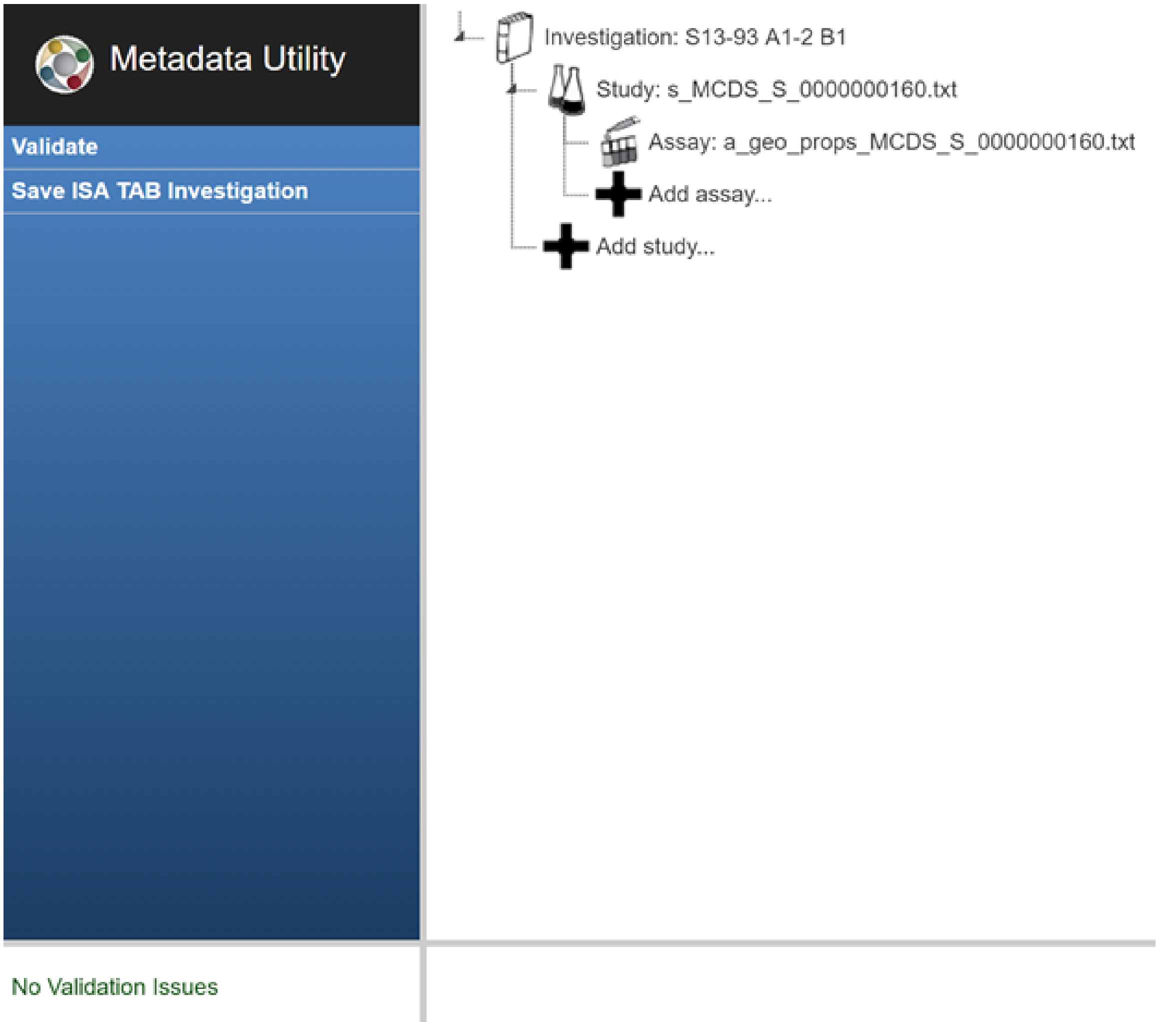
No Validation Issues



Investigation: S13-93 A1-16 B

Study: s_MCDS_S_0000000150.txt

Assay: a_geo_props_MCDS_S_0000000150.txt



Investigation: S13-93 A1-2 B1

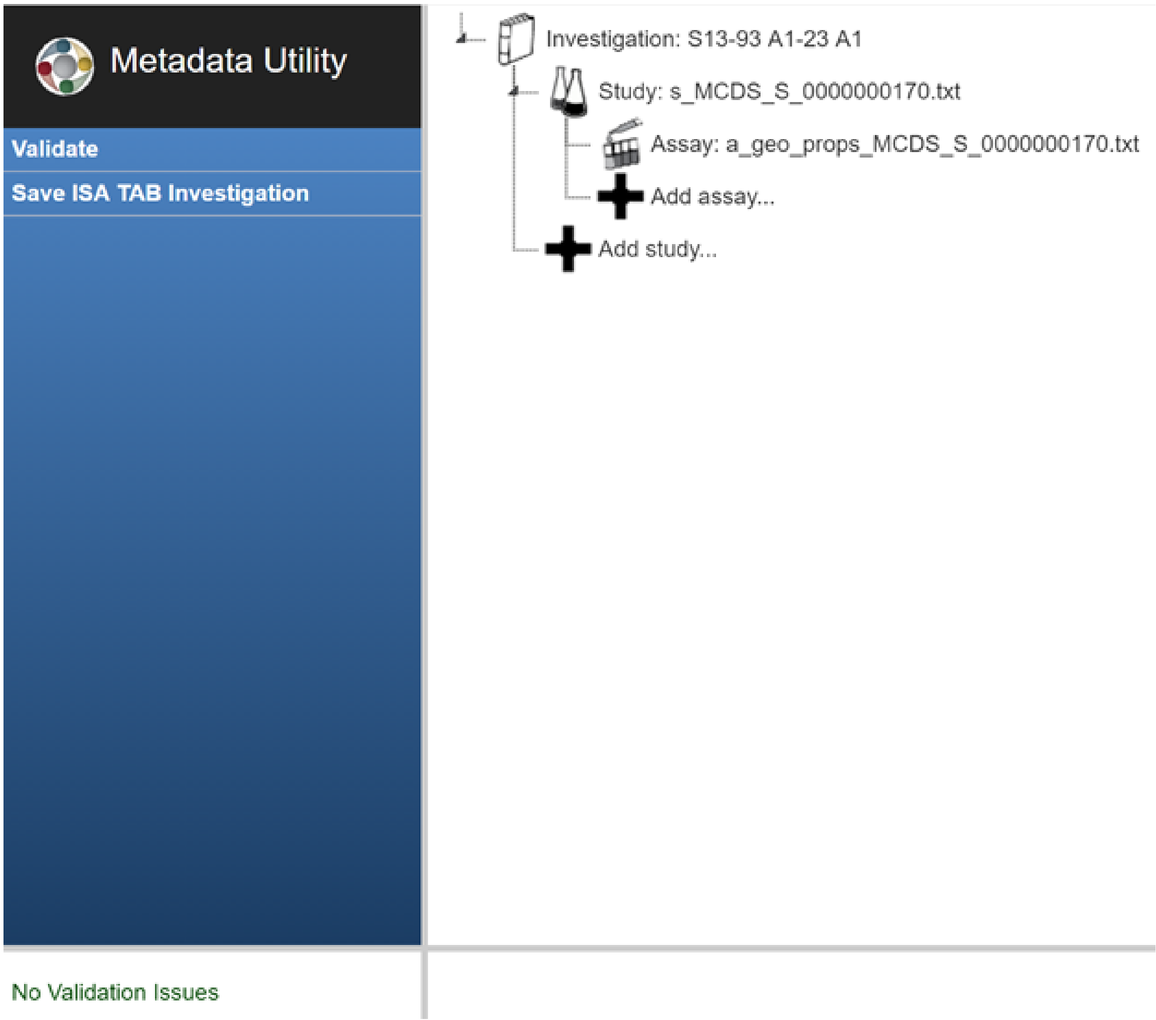
Study: s_MCDS_S_0000000160.txt

Assay: a_geo_props_MCDS_S_0000000160.txt

Add assay...

Add study...

No Validation Issues

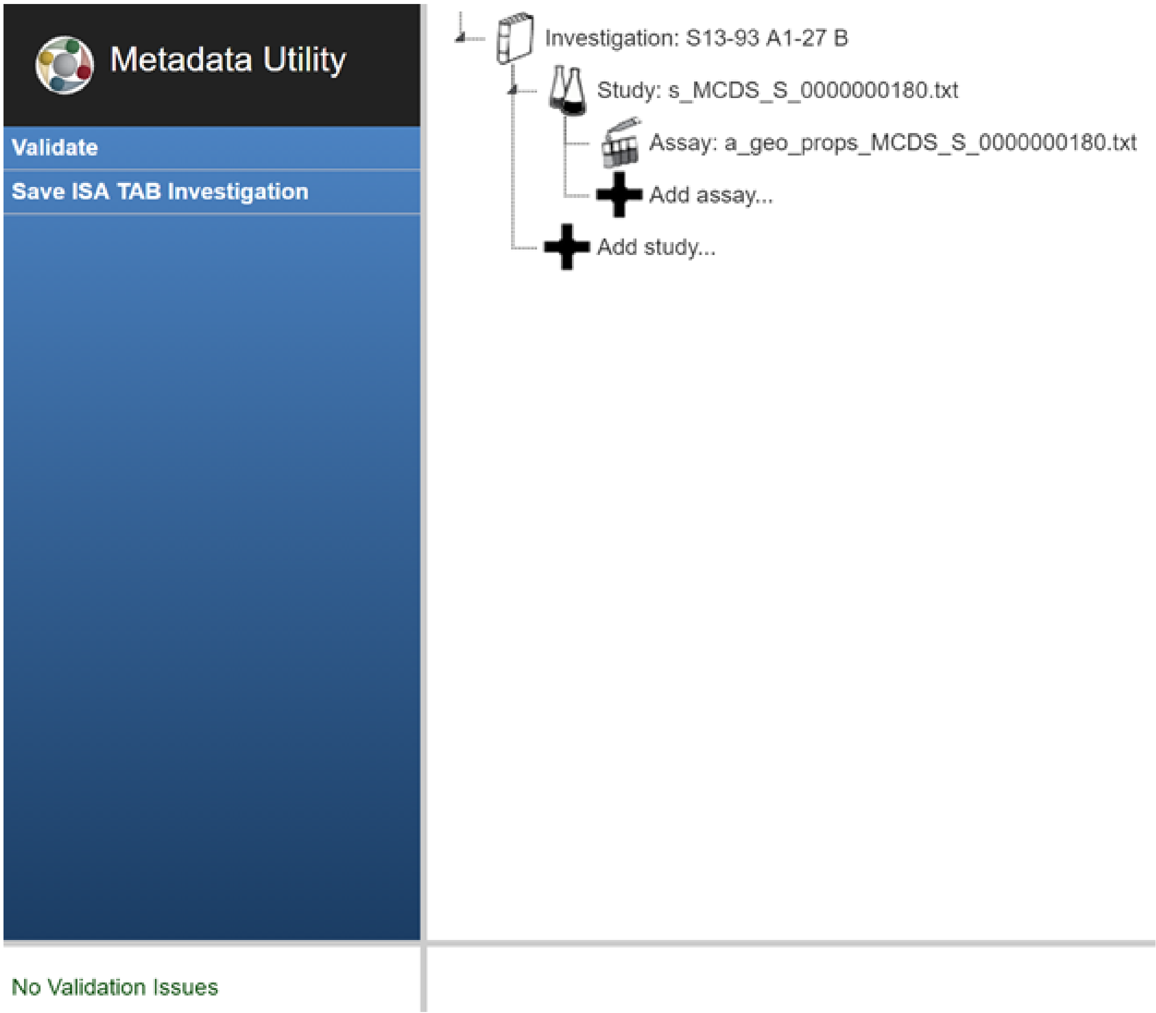


Investigation: S13-93 A1-23 A1

Study: s_MCDS_S_0000000170.txt

Assay: a_geo_props_MCDS_S_0000000170.txt

No Validation Issues



Investigation: S13-93 A1-27 B

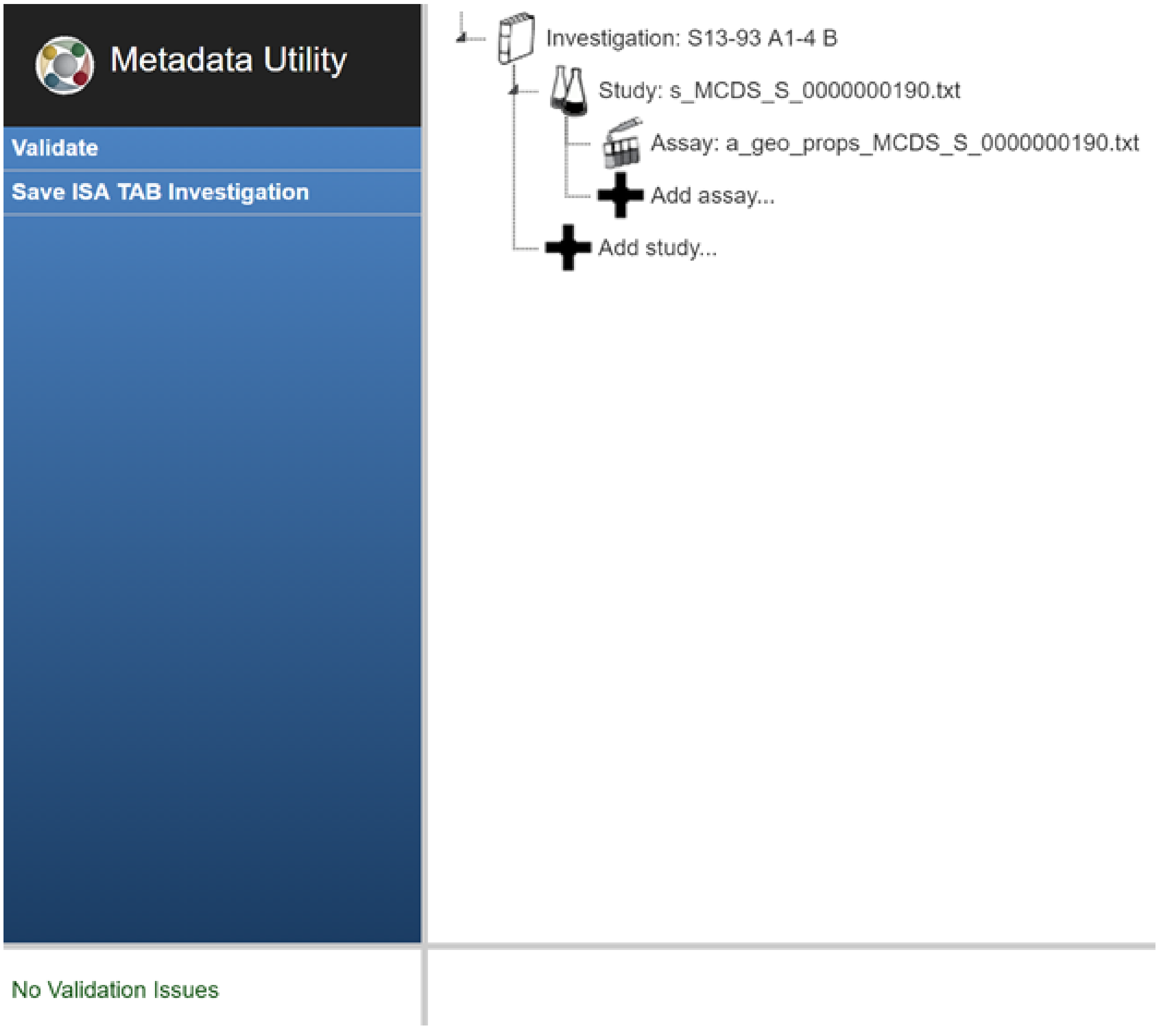
Study: s_MCDS_S_0000000180.txt

Assay: a_geo_props_MCDS_S_0000000180.txt

Add assay...

Add study...

No Validation Issues

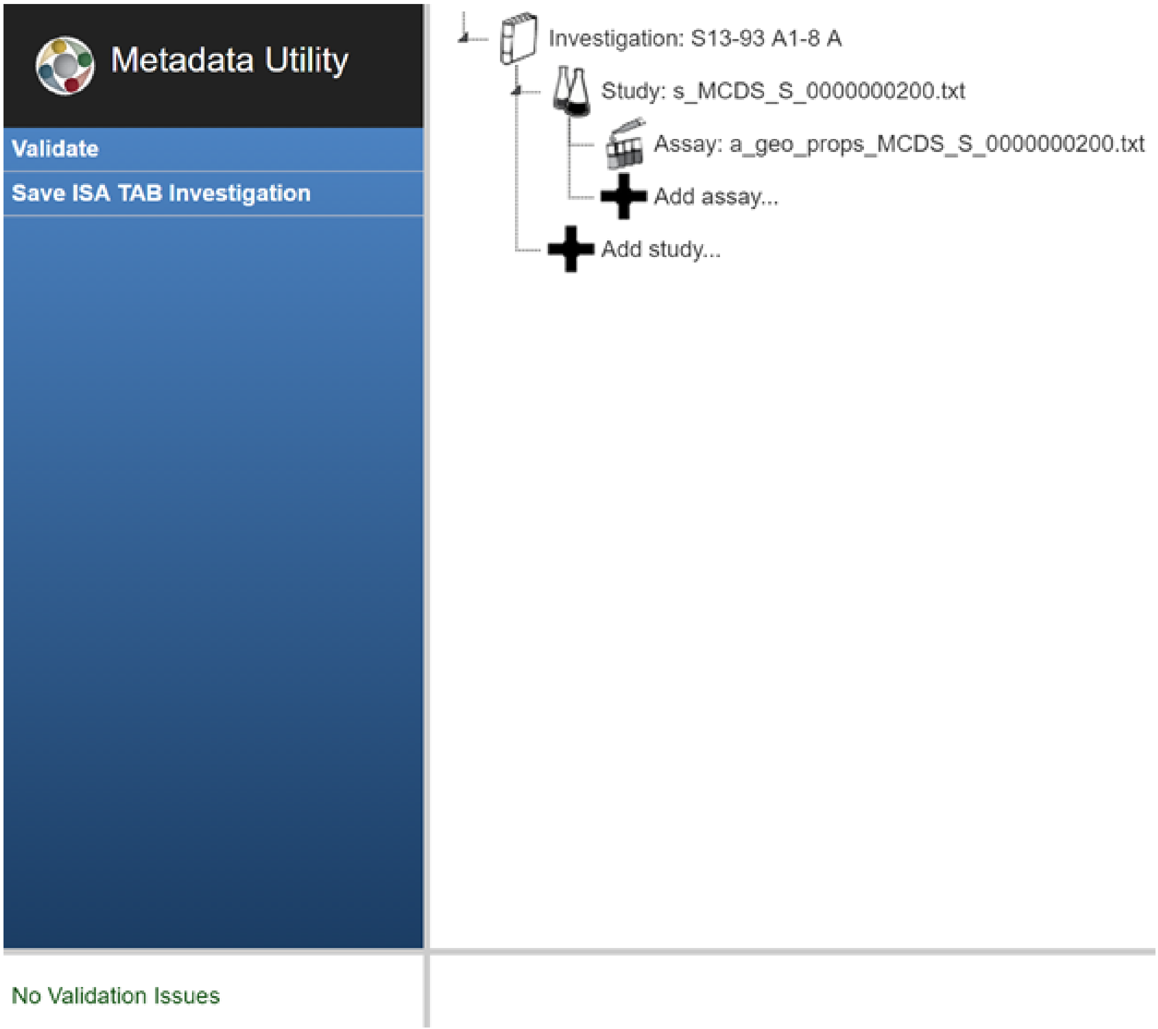


Investigation: S13-93 A1-4 B

Study: s_MCDS_S_0000000190.txt

Assay: a_geo_props_MCDS_S_0000000190.txt

No Validation Issues



Investigation: S13-93 A1-8 A

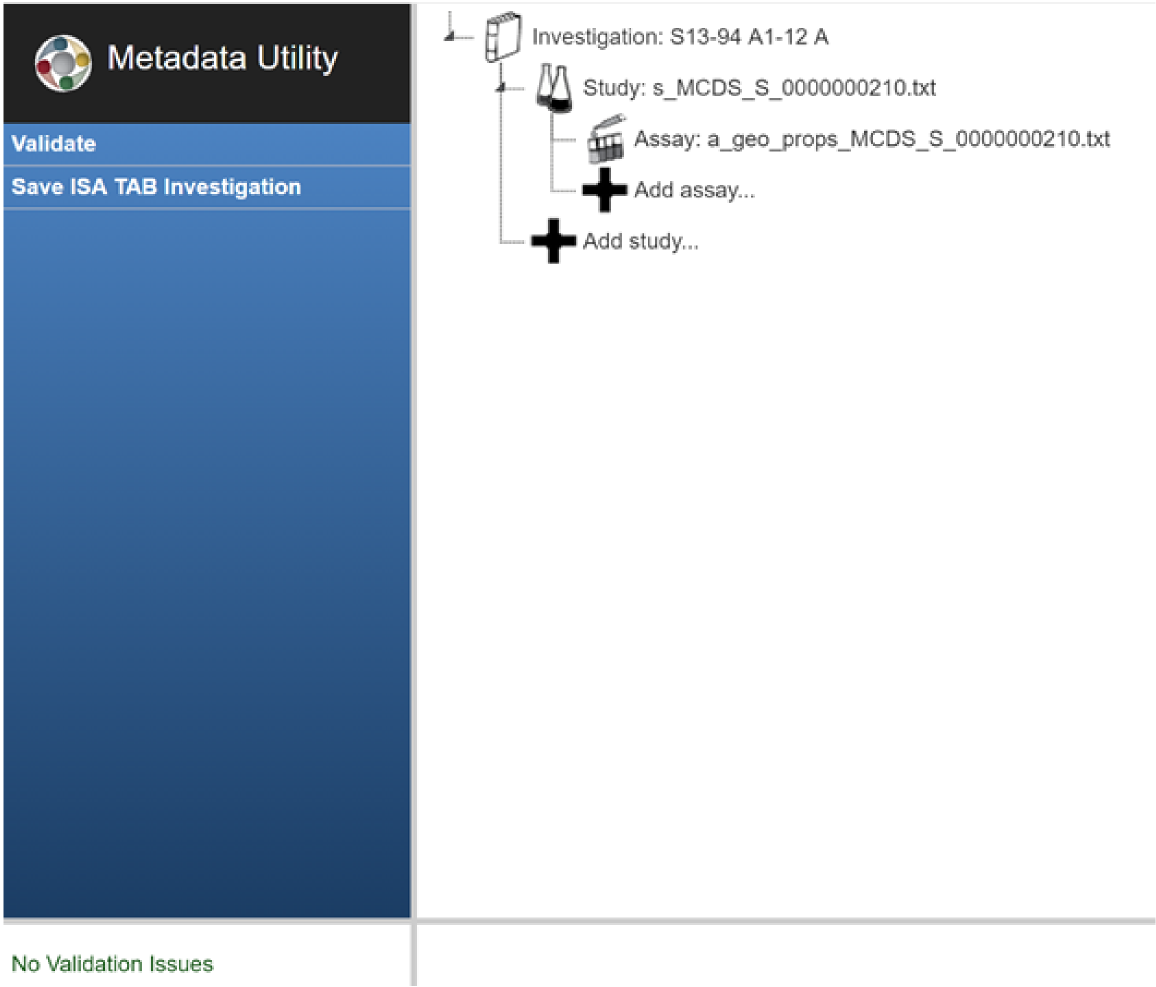
Study: s_MCDS_S_0000000200.txt

Assay: a_geo_props_MCDS_S_0000000200.txt

Add assay...

Add study...

No Validation Issues

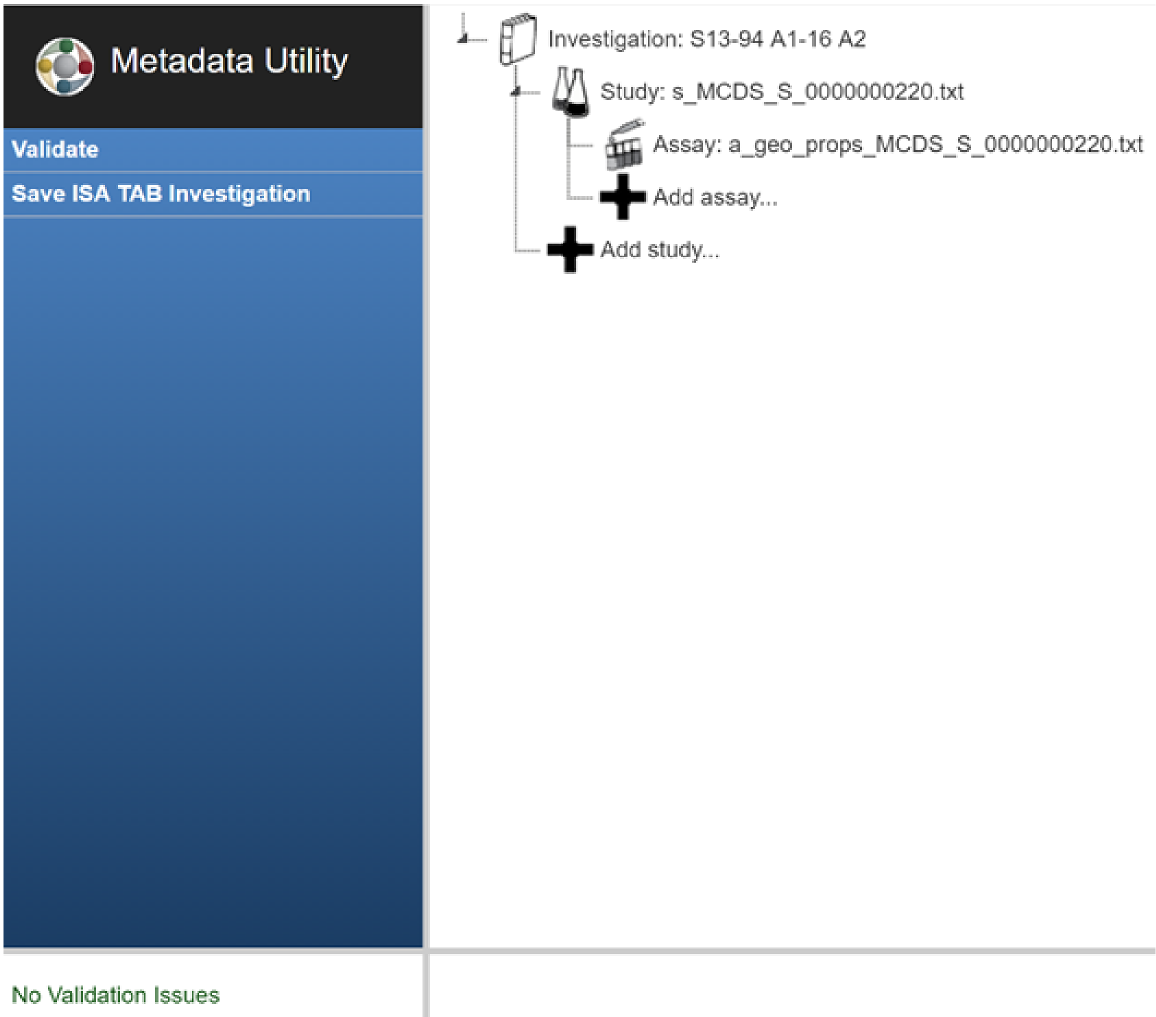


Investigation: S13-94 A1-12 A

Study: s_MCDS_S_0000000210.txt

Assay: a_geo_props_MCDS_S_0000000210.txt

No Validation Issues

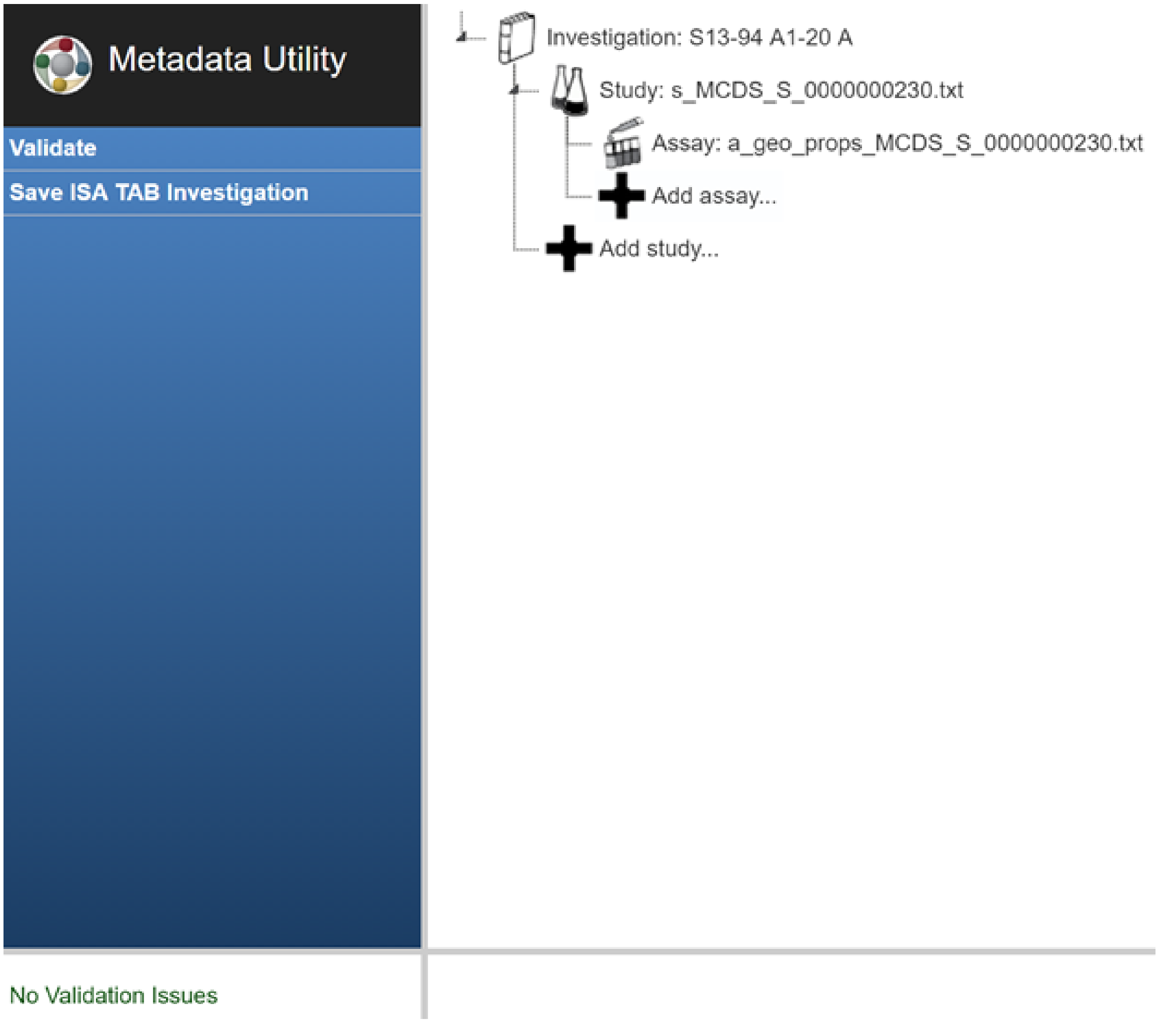


Investigation: S13-94 A1-16 A2

Study: s_MCDS_S_0000000220.txt

Assay: a_geo_props_MCDS_S_0000000220.txt

No Validation Issues

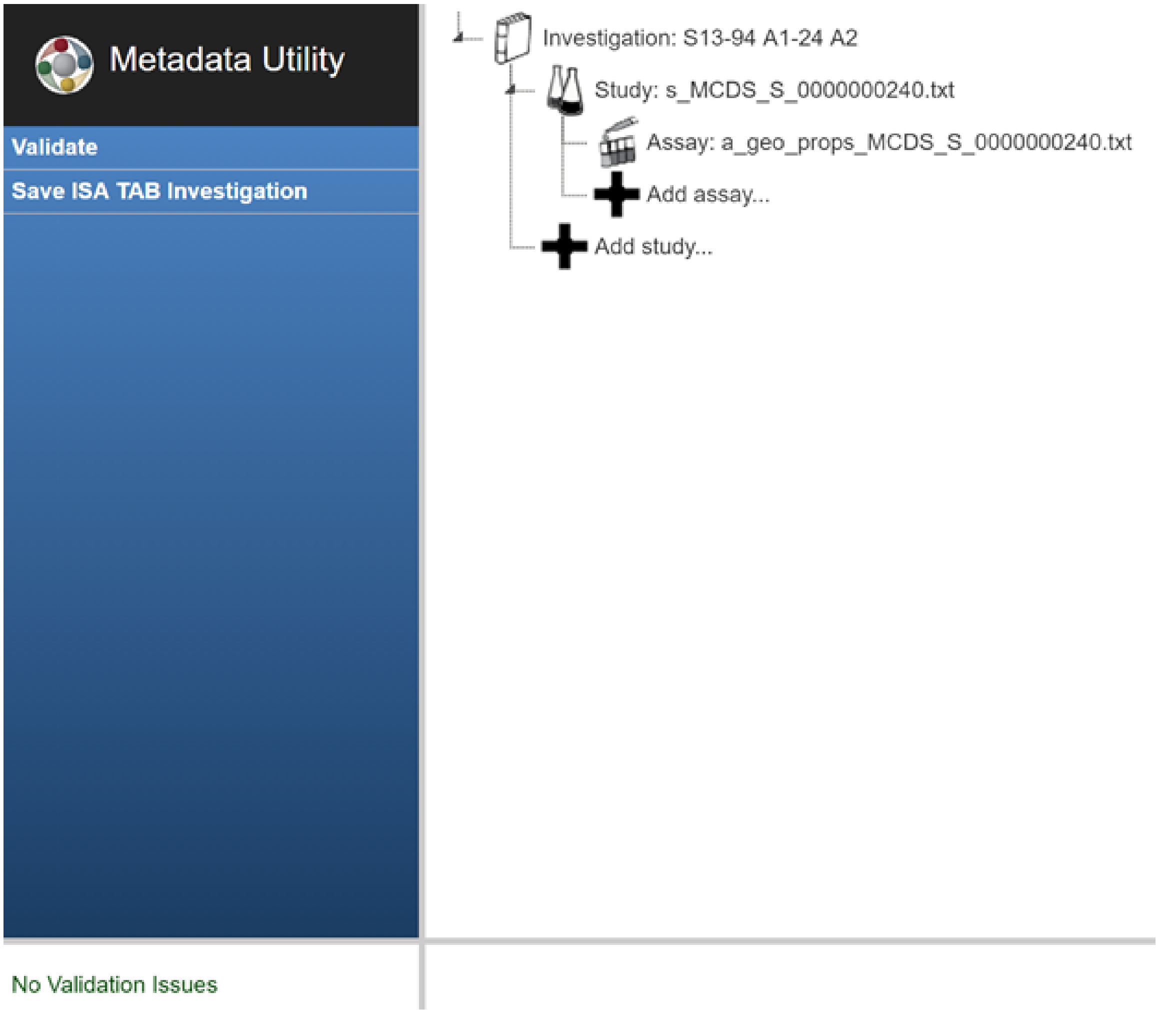


Investigation: S13-94 A1-20 A

Study: s_MCDS_S_0000000230.txt

Assay: a_geo_props_MCDS_S_0000000230.txt

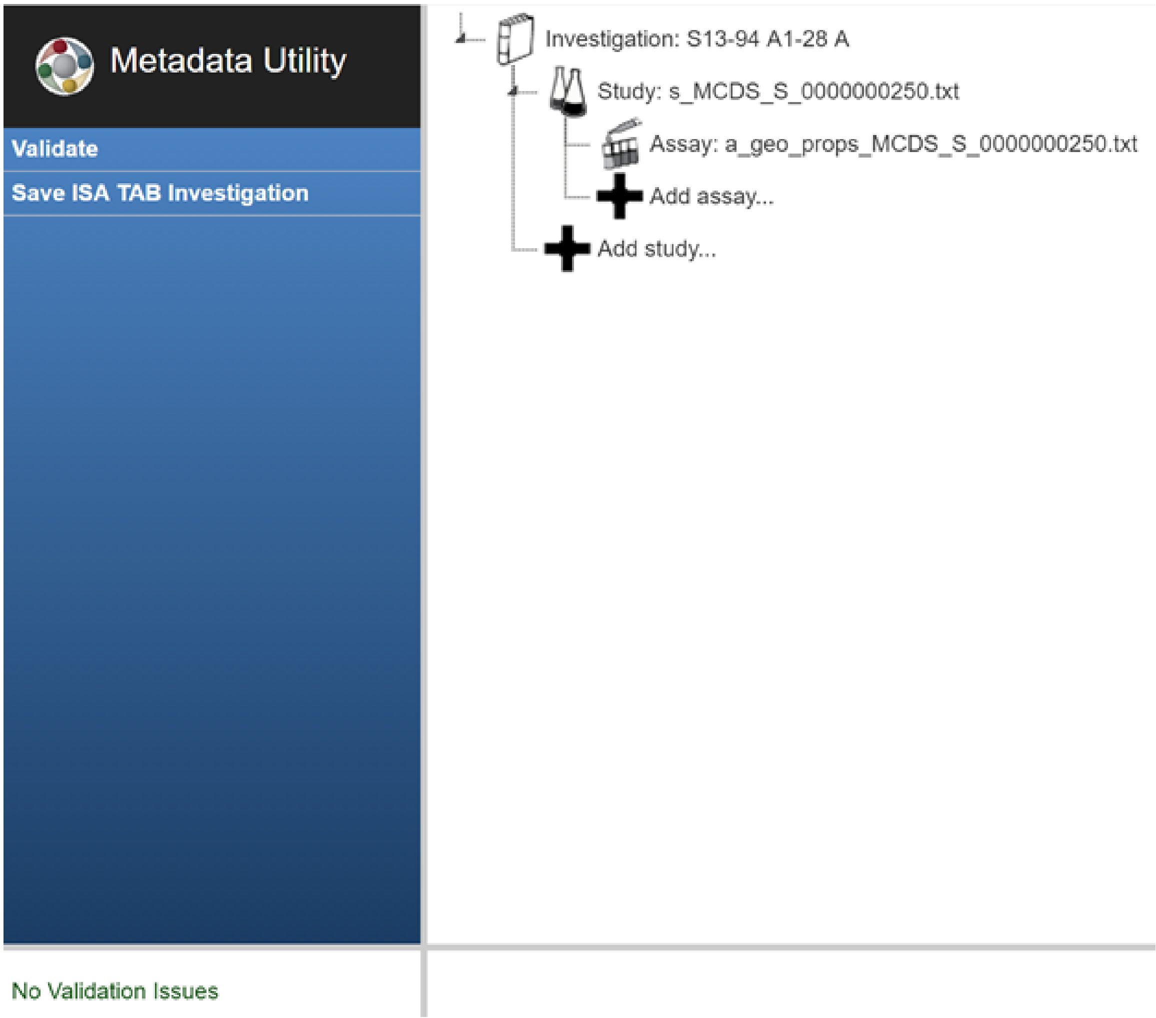
No Validation Issues



Investigation: S13-94 A1-24 A2

Study: s_MCDS_S_0000000240.txt

Assay: a_geo_props_MCDS_S_0000000240.txt

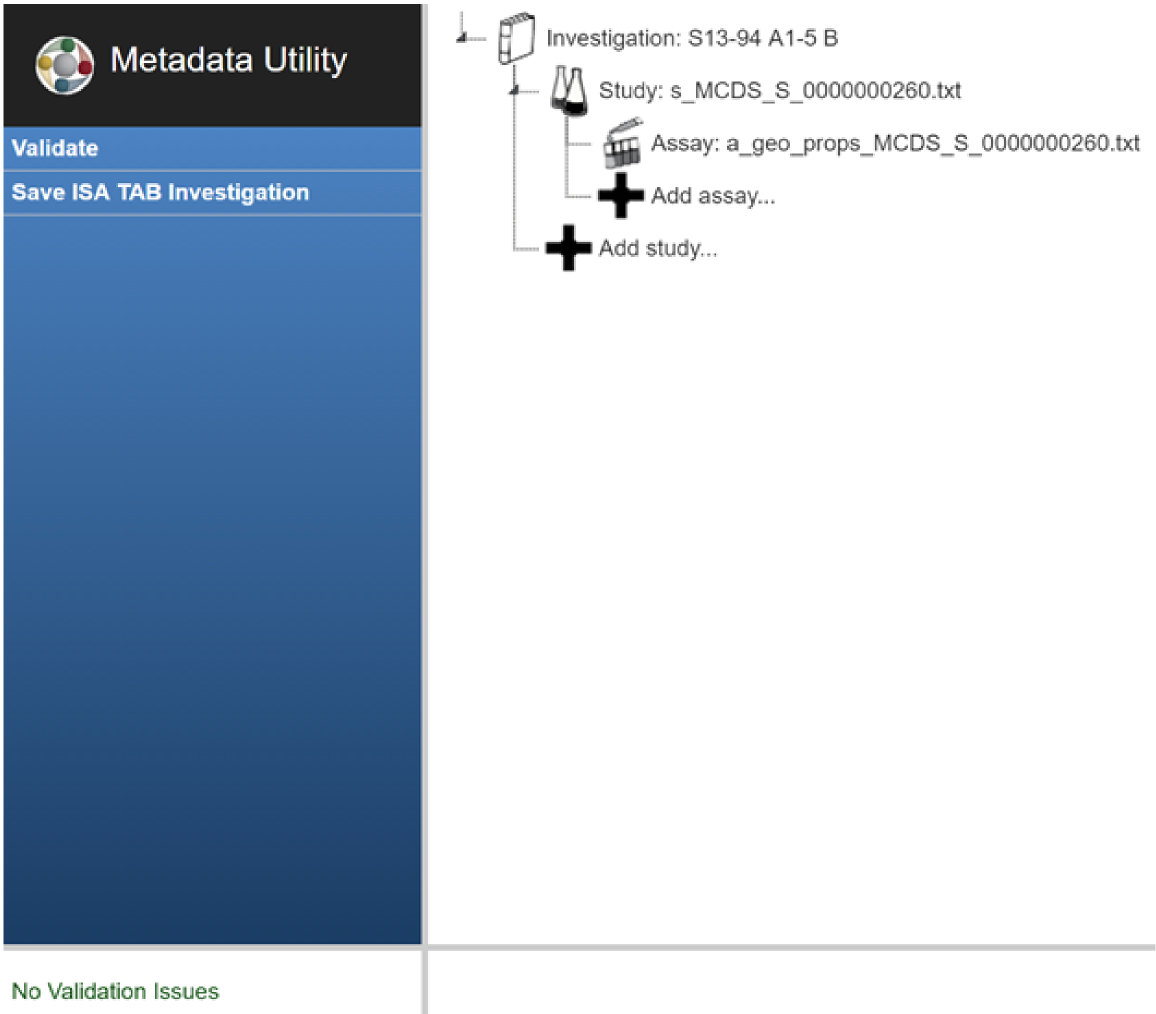


Investigation: S13-94 A1-28 A

Study: s_MCDS_S_0000000250.txt

Assay: a_geo_props_MCDS_S_0000000250.txt

No Validation Issues



Investigation: S13-94 A1-5 B

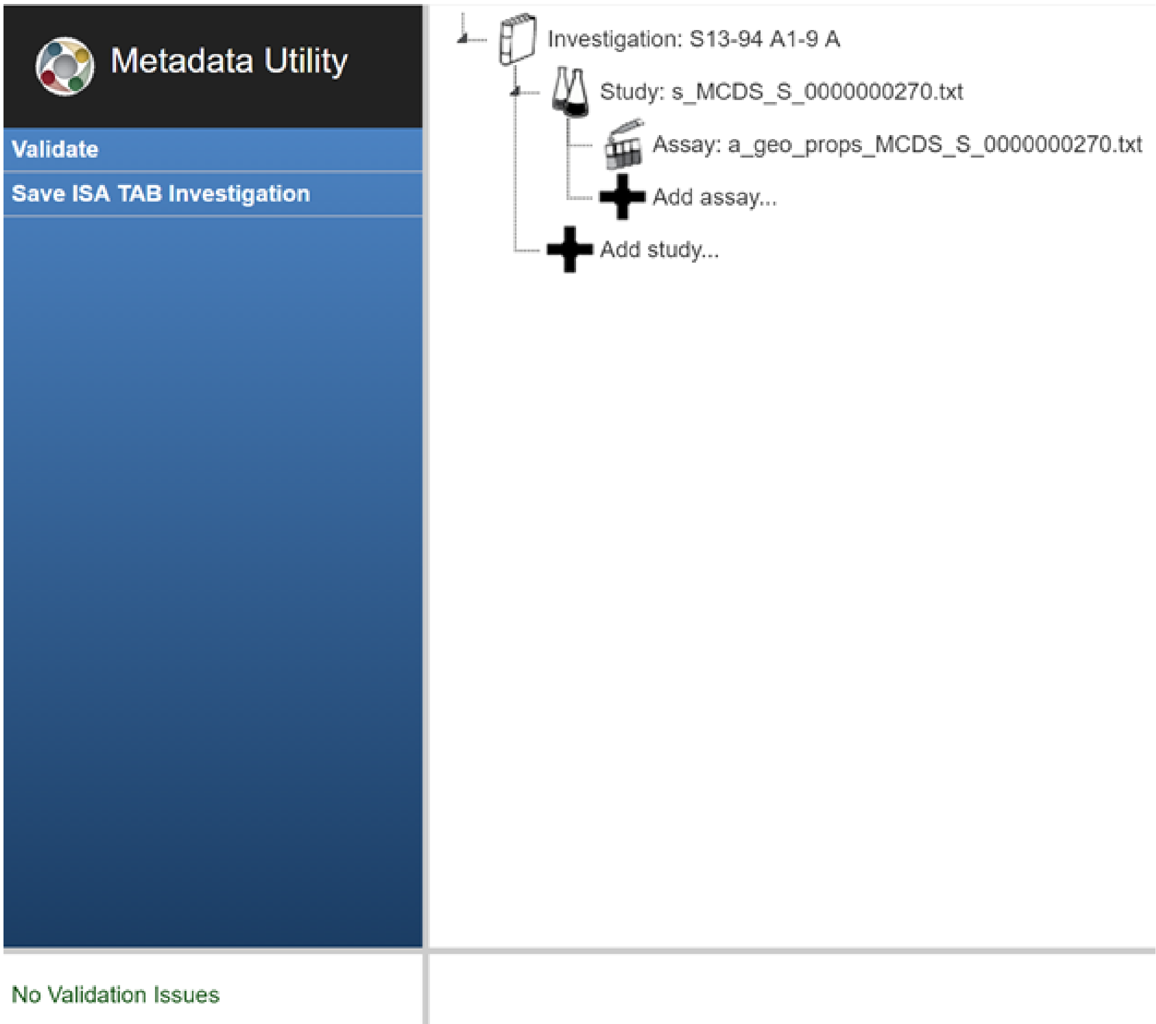
Study: s_MCDS_S_0000000260.txt

Assay: a_geo_props_MCDS_S_0000000260.txt

Add assay...

Add study...

No Validation Issues

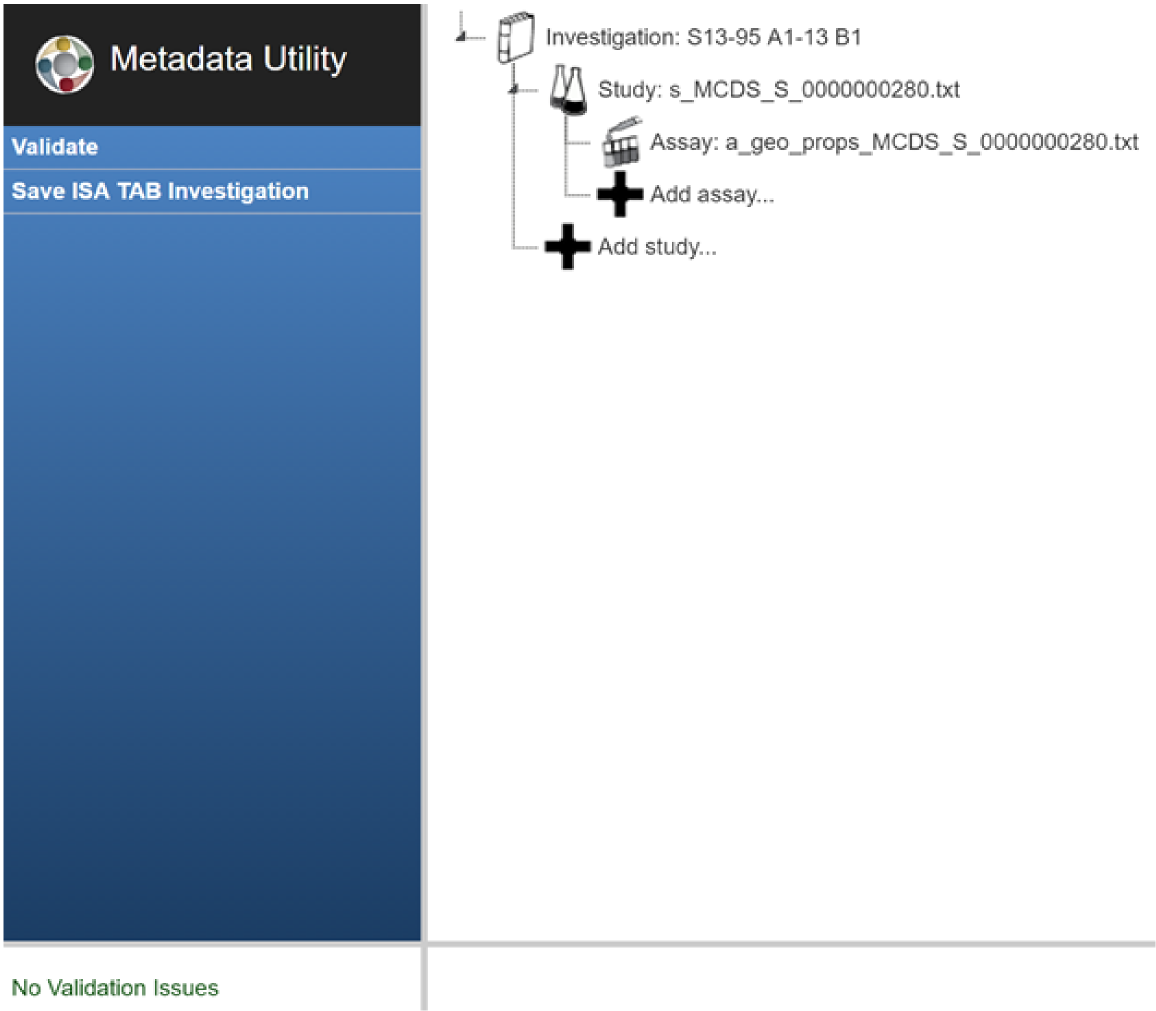


Investigation: S13-94 A1-9 A

Study: s_MCDS_S_0000000270.txt

Assay: a_geo_props_MCDS_S_0000000270.txt

No Validation Issues

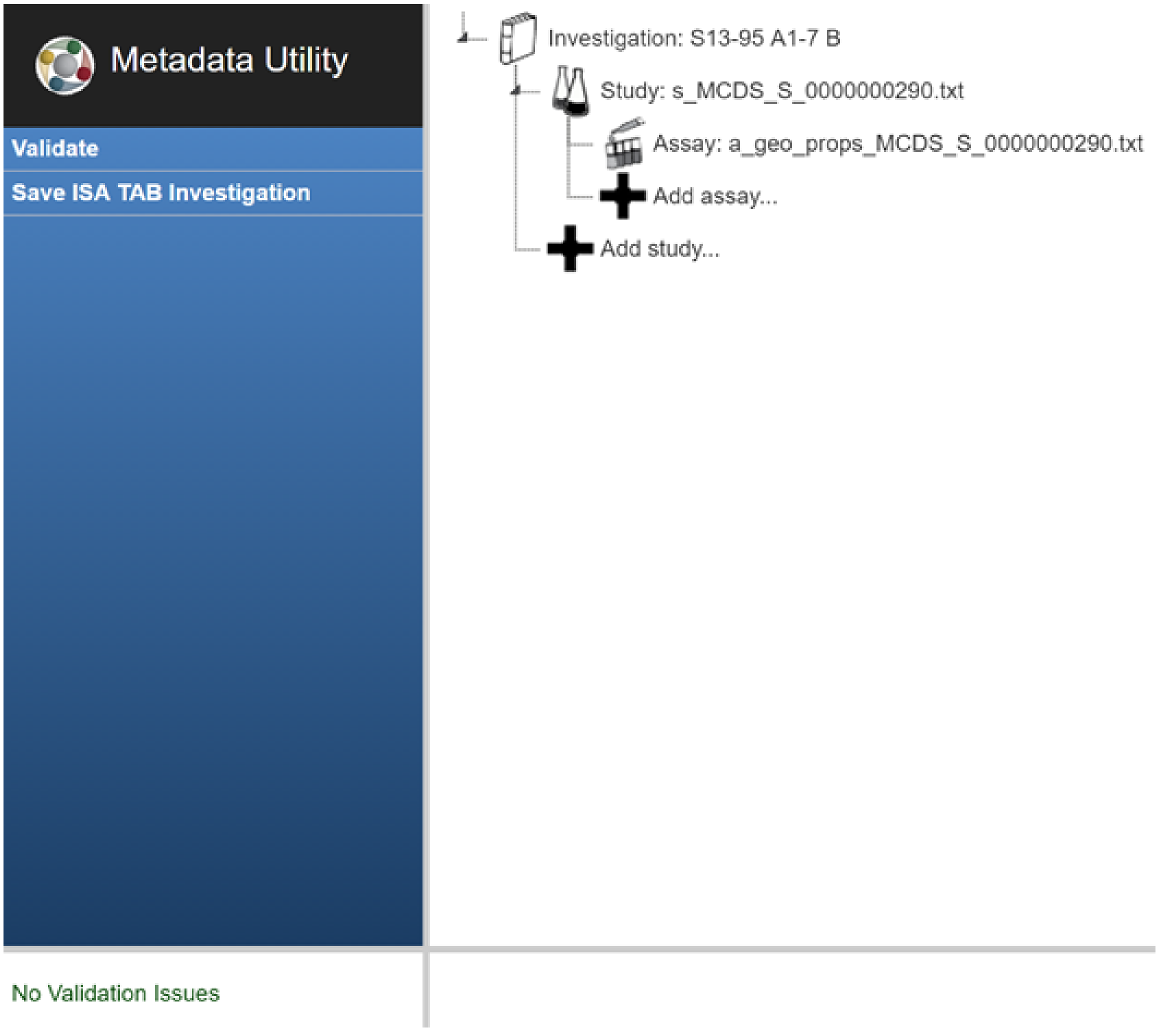


Investigation: S13-95 A1-13 B1

Study: s_MCDS_S_0000000280.txt

Assay: a_geo_props_MCDS_S_0000000280.txt

No Validation Issues

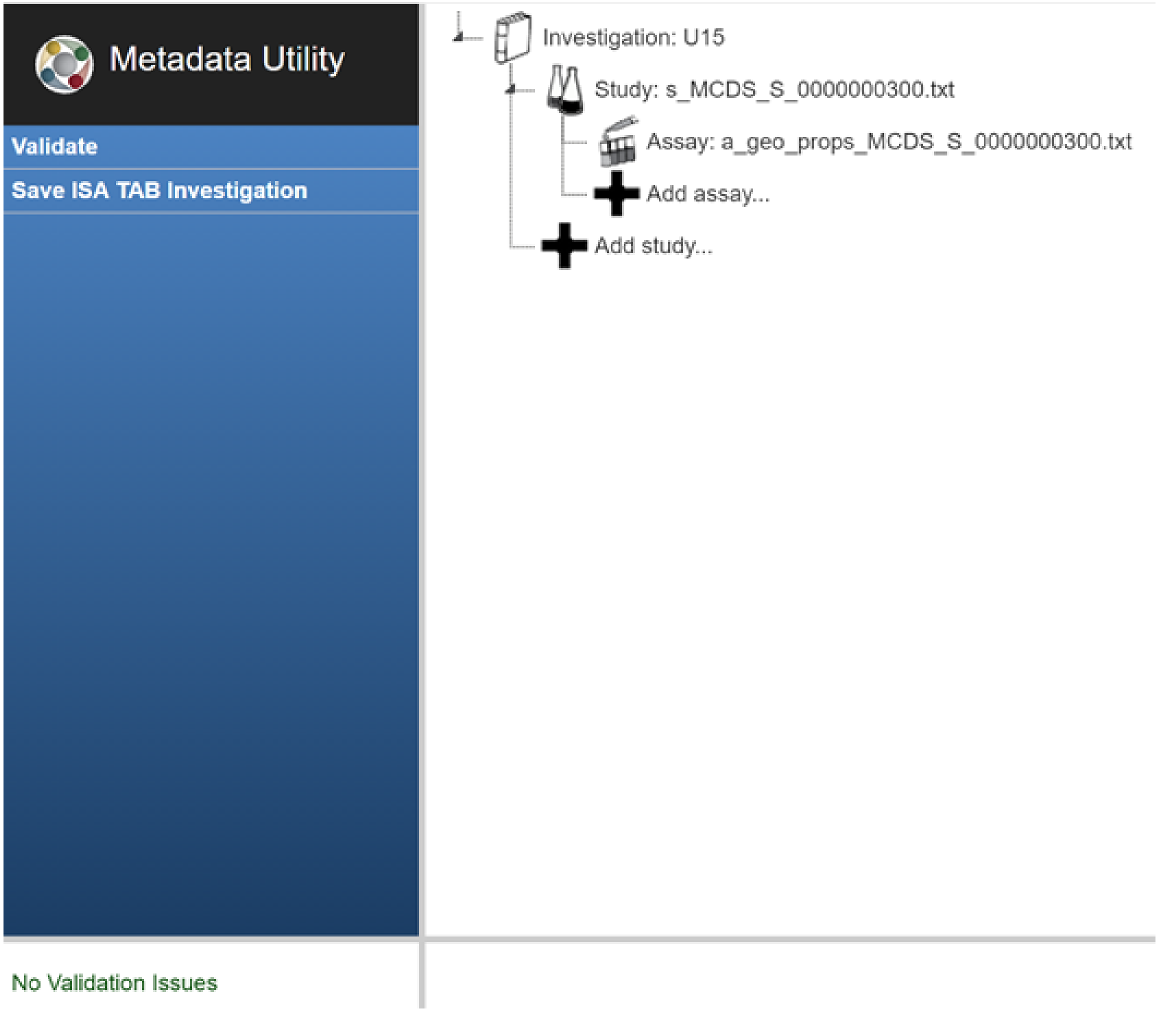


Investigation: S13-95 A1-7 B

Study: s_MCDS_S_0000000290.txt

Assay: a_geo_props_MCDS_S_0000000290.txt

No Validation Issues



Investigation: U15

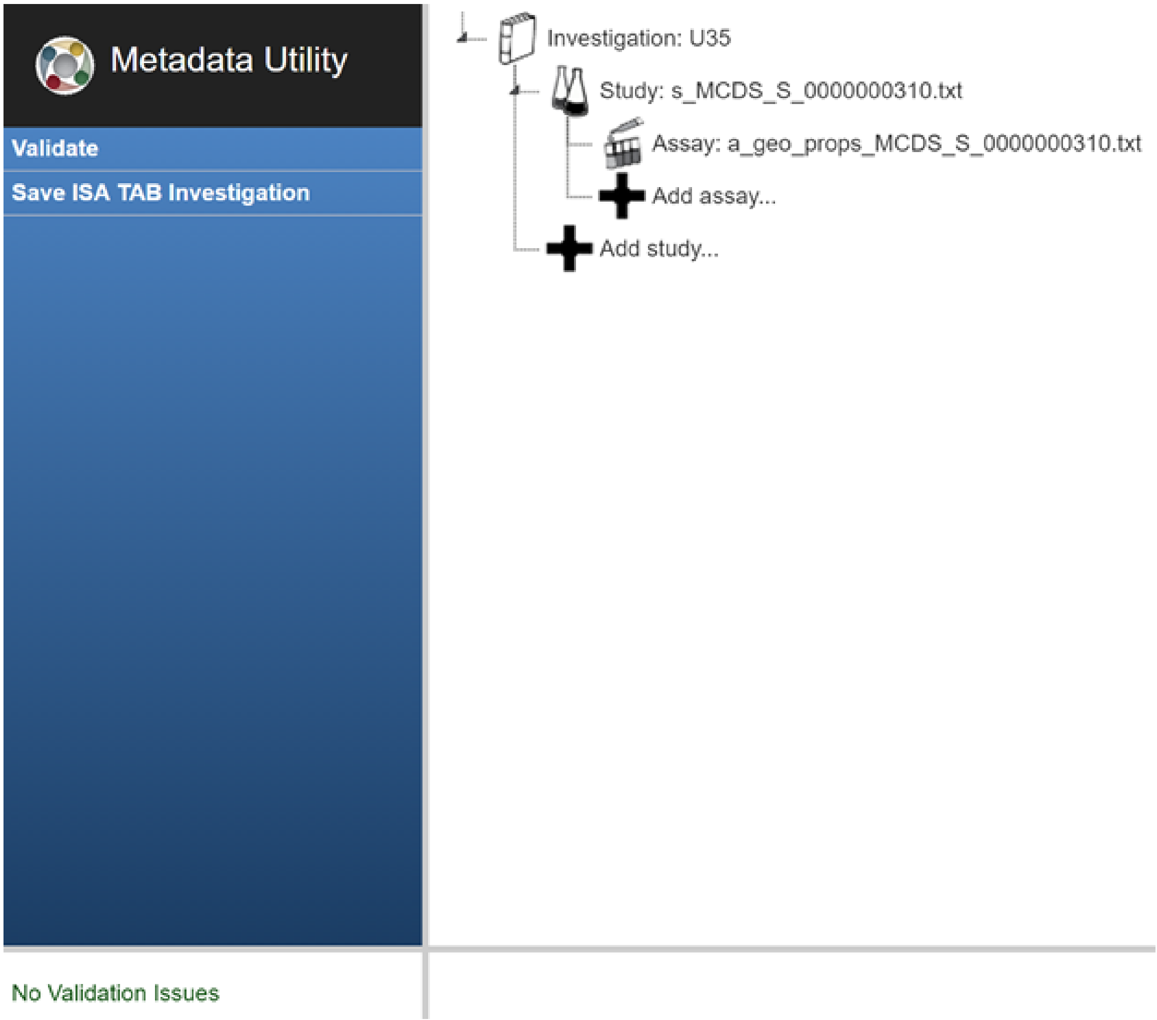
Study: s_MCDS_S_0000000300.txt

Assay: a_geo_props_MCDS_S_0000000300.txt

Add assay...

Add study...

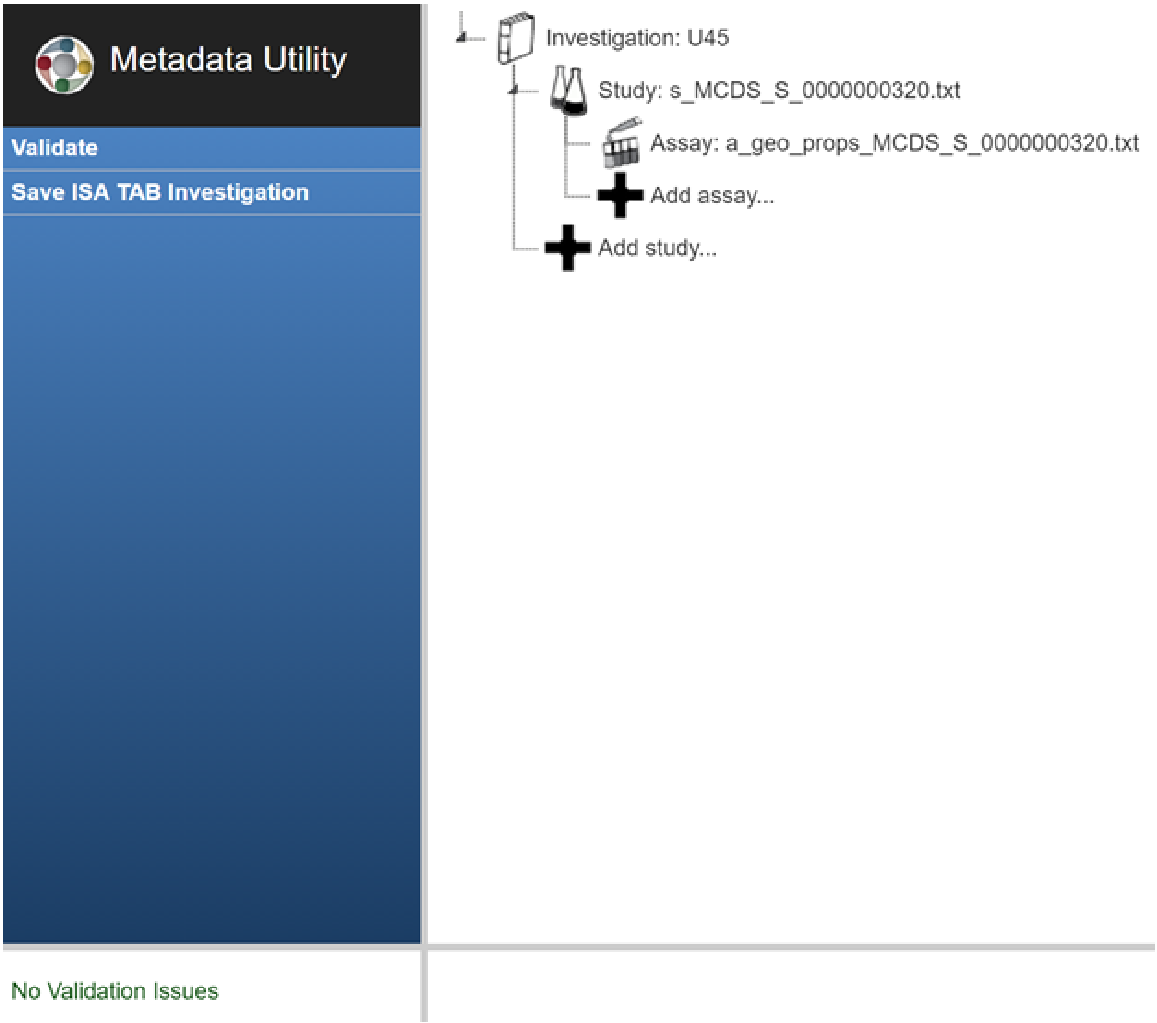
No Validation Issues



Investigation: U35

Study: s_MCDS_S_0000000310.txt

Assay: a_geo_props_MCDS_S_0000000310.txt

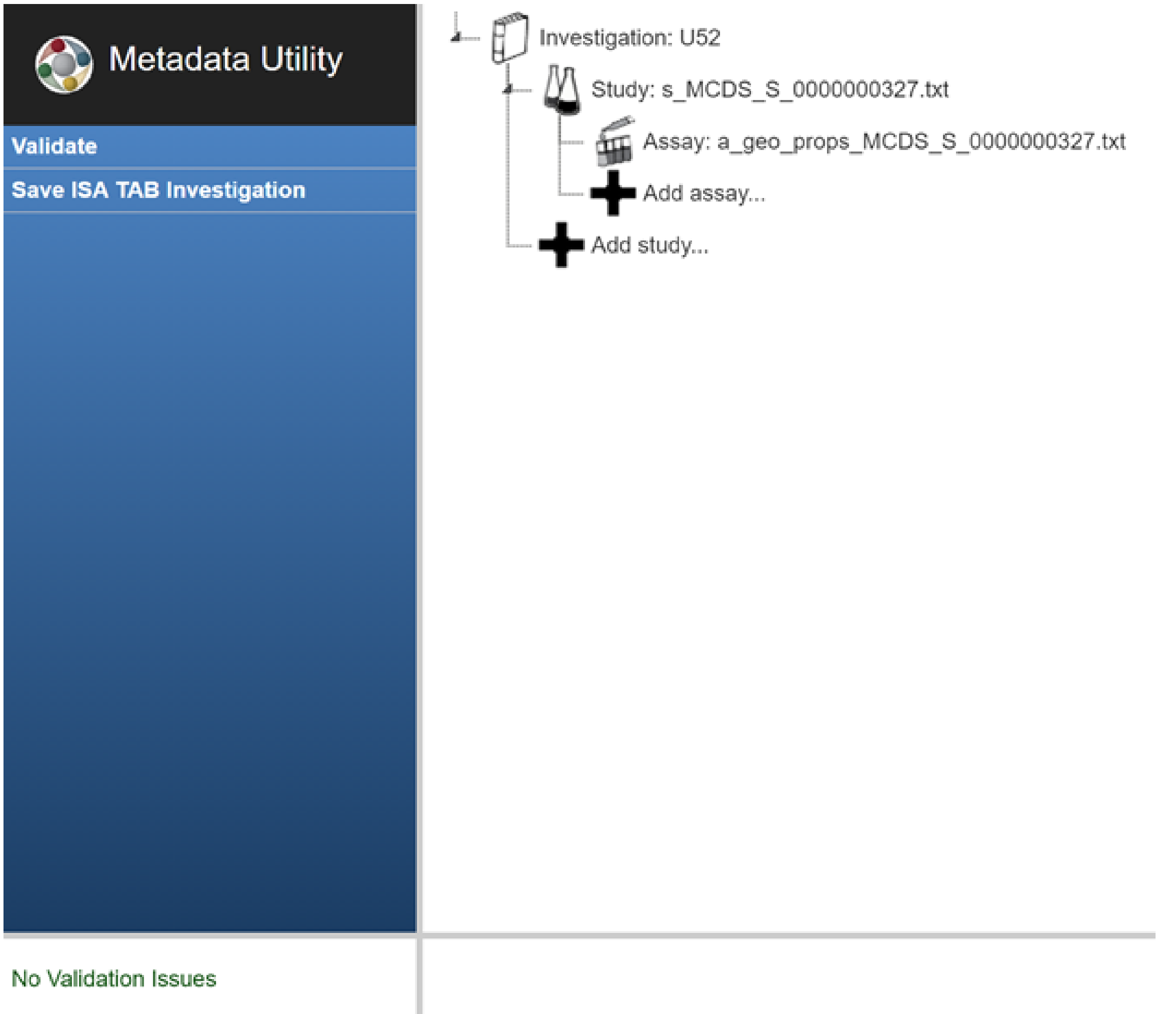


Investigation: U45

Study: s_MCDS_S_0000000320.txt

Assay: a_geo_props_MCDS_S_0000000320.txt

No Validation Issues



Investigation: U52

Study: s_MCDS_S_0000000327.txt

Assay: a_geo_props_MCDS_S_0000000327.txt

No Validation Issues

VITA

Corey Peyton Chitwood graduated with a Bachelor’s of Science in Bioengineering from the University of Louisville in May 2020 and Master’s of Engineering in Bioengineering from the University of Louisville in July 2021. At the University of Louisville, he was a recipient of the Henry Vogt Scholarship. Corey has previously worked in the Clinical Engineering Department at Cincinnati Children’s Hospital, and plans to apply his experience and passion for medicine in to a career as a Physician in the future.