Data Persistence and Access Architecture for Open and Reproducible Science

by

Wen Ting (Maria) Tu

B.A.Sc., University of British Columbia, 2020

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Wen Ting (Maria) Tu**
Degree:	**Master of Applied Science**
Thesis title:	**Data Persistence and Access Architecture for Open and Reproducible Science**
Committee:	**Chair:** Rodney Vaughan Professor, Engineering Science
	Stephen Makonin Co-Supervisor Adjunct Professor, Engineering Science
	Ivan V. Bajić Co-Supervisor Professor, Engineering Science
	Chinthaka Pathum Dinesh Committee Member Adjunct Professor, Engineering Science
	Fred Popowich Examiner Professor, Computing Science

Abstract

Reproducible science benefits the research community by providing users with credible data, which promotes accelerated progress in methodological research and innovation. This work proposes a preliminary method using persistent identifiers (PIDs) to reproduce research results using time-series data. PID is a long-lasting reference that allows specific data to be queried and accessed from a request originating from outside the system. A PID prevents unintentional alteration or corruption. The methodology and design for the prototype are proposed and built based on the IEEE Standards Association P2957 Data Governance and Metadata Management Working Group (BDGMMWG). A prototype is presented that implements parts of the Federated Metadata Registry diagram from BDGMMWG. Datasets are used to demonstrate (i.e., the prototype) how the proposed PID system architecture can be used for time-series data to create persistent links to specific data within a dataset. Providing such links are key to retrieving the data used to reproduce scientific experiments and verifying published results.

Keywords: Reproducibility; Repeatability of Scientific Results; PID; ARK, time-series datasets; IEEE Standards

Acknowledgements

Many thanks to my supervisor Stephen Makonin for meeting regularly and offering great suggestions and feedback on the progress of writing my thesis. Many thanks to my family, especially my mother, for the great support. Many thanks to Collin for patiently listening to me and supporting me along the way. Many thanks to my friends for being here for me. Many thanks to Professor Bajić and Professor Dinesh for being on the committee.

I would also like to thank the IEEE Standards Association for allowing me to participate in the creation of the P2957 white paper. Special thanks to John Kunze of ARKs and the Ronin Institute for his valuable feedback and suggestions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Reproducible Science

Reproducible science and reproducible research provide the opportunity for researchers to generate the same results given the original data and procedures used for the published experiment [24]. As a result, reproducible science benefits every single researcher in the scientific community because it creates credibility amongst published works. If a researcher decides to perform experiments on the existing results, or extend or improve on research similar to the ones that existed, then reproducible science permits such actions to continue without worrying about the consequence of unreliable data. However, there are concerns, which we discuss below.

One concern among the scientific community is the incidence of scientific misconduct, including fabrication, falsification, and plagiarism. In the field of reproducible science, fabrication and falsification are the two main concerns. Fabrication refers to the usage of synthetic data or fictitious results to achieve the desired experimental results; and falsification is manipulating research materials, procedures, or altering or omitting recorded data or results so that the experimental results look desirable [25].

In 1981, John Roland Darsee, who was working at Harvard Medical School, was suspected by several researchers that he was systemically making up a major part of his research to achieve an astonishing outcome [26]. Harvard took public action six months after the admitted falsification and explained publicly that the problems had to do with Darsee and his inability to produce certain critical raw data [26]. Not only is the process of getting testified long and involves a lot of manpower, but it also slows down the study of other colleagues working alongside, in need of the results for their own experiment. In addition, the subsequent research using John Roland Darsee's data is also impacted and could result in the retraction of already published papers. The event would have resulted in many researchers losing confidence in using other researchers' data due to the distrust about the credibility.

Having a reliable platform that can provide data in one central location will shorten peer review time. Given the published research procedures and data, other researchers could easily replicate the same experiment. If enough researchers fail to achieve the same results as the original researcher, then the publication will become unreliable, which could impact the original researcher's reputation unless further explanations are made. In order to prevent reputation harm in the scientific field, researchers will be more aware of the validity of their publication, which, as a result, will greatly reduce the likelihood of fabrication and falsification.

When research is verifiable, concerns about using the results of the paper diminish, which promotes a good connection network of researchers sharing and using each other's work and data. Many researchers can verify the publication when they are replicating the same experiment, provided that the procedures and experimental data are stored in one location. Faster access for data retrieval would result in a faster verification process which would also accelerate subsequent experiment publication building on the existing results. Given one central location for data storage and if the results have been authenticated to be unreliable, the retracting process can also be fast and easy. When verified data, documents, and results are presented to the public, accelerated progress is achieved in methodological research and innovation because more scientists have access to information for their research [24]. Verified reliable data allow peers to understand and reproduce each other's work. Reconfiguration of previously conducted experiments with similar tasks becomes easier and more adaptive, which increases the quality and speed of peer review [27].

There are limitations on resources and time allotment that can prevent an experiment from being carried out. However, if the validity of the data used in the scientific paper is already verified, the results of the experiment, and the data are made available, subsequent experiments from different facilities or institutions can be made. The running and publishing of these experiments promotes a faster peer-review process.

In reproducible science, data, code, and procedures are documented, and the results of the study or the computations can be executed again with identical results. The assumption is that applying the same methodological analysis on the same used data would produce the same results [28].

The International Data Corporation remarked that the worldwide volume of data doubles every 18 months [6]. The number of world data will increase from 33 zettabytes recorded in 2018 to an astonishing number of 175 zettabytes by 2025 [29]. Given the increasing amount of availability of data that is made public and the readily available affordable computer resources for computation, reproducible science helps evaluate and maintain the stored data at a consistent standard across the field. The platform will maintain the data at a high standard and provide continuous access which would reduce the work of the researchers. Consequently, researchers can focus their time on the research and devote their energy to constructing a more detailed description of the procedures and data used.

As mentioned in the IEEE Big Data Governance and Metadata Management Standards Roadmap White Paper, there is an increase in the need for quality data to satisfy the growing demands for big data analytics and Artificial intelligence algorithms [29]. Understanding the data landscape is important to support the upcoming growth of AI challenges and big-scale data analytics [29]. As the growth of artificial intelligence algorithms, such as machine learning and deep learning algorithms, becomes more complex, lack of specificity as to what data were used for training and testing these learning systems becomes a major problem. Many scientific papers present concise descriptions, making it difficult to understand the specific data and the exact procedures to follow. Training with new independent data for comparison may not be possible. To replicate the experiments, exact data should be followed to eliminate such problems.

There is an enormous amount of data called "Big Data" which is accumulated and collected via the rapid expansion of network computing systems [27]. Data is often collected into a single database without a specific data mining task in mind. Nevertheless, in real-world applications, there are always trivial or meaningless data called noise that make up some part of the data and do not contribute to the analysis of the data [30]. If the data is collected into a database that is built with a purpose and has additional features such as data processing; hence, when there is selected specificity on the data, the quality of the data can be reassured. Therefore, the action of choosing the right data before conducting the experiment becomes extremely important. One method such as the Knowledge Discovery in Databases helps with this problem.

Knowledge Discovery in Databases (KDD) is used to extract valuable information from the vast amount of data offered to the public by understanding the data and recognising its pattern [6]. Figure 1.1 shows an overview of the KDD process in detail [6]. It gives out the different procedures and process terms in detail.

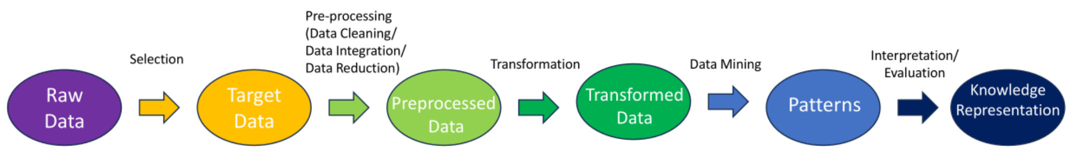


Figure 1.1: Overview of the KDD process. Adapted from [6]

Reproducible science allows researchers working in the same field to be able to access data and code to carry out further experiments on existing results without worrying about the validity and accountability of the past results. With the same application of scientific analysis on the same usage of raw data conducted by the original researcher, the same experimental result would be formed by researchers trying to conduct the same experiment [31].

1.2 Difficulty in Finding Data for Reproducibility

The reproducibility crisis occurs when some scientific studies or results are demanding and are almost impossible to reproduce. Some experiments in specific areas of study require extensive computing power, which would not be accessible to all researchers [32]. In some other experiments, specific permits from the government are required. In areas such as environmental epidemiology, experiments are conducted on specific objects over long periods of time, which is expensive and difficult to replicate when results are needed in time for policy regulation decisions [32]. Therefore, if researchers can access prior knowledge or data in this area, the study can be processed faster. However, in order to do that, reliable research results are needed.

When the validity and credibility of the research results are unreliable, it is difficult to build a subsequent experiment on top of existing results [33]. Therefore, in order to minimise this problem, finding and storing qualified data at high standards becomes essential in the field of reproducible science. When the data are achieving a high level of standard, the researcher should be able to use the same data and apply the same experimental procedure to achieve the same result every single time.

To reproduce the original experiments, researchers need access to the same stored data and apply identical methodologies. Lack of access to systems such as data repositories should be granted in order to promote data sharing and reduce the issue of having difficulty in finding data for reporducibility [34]. However, there are many factors that affect the ability of researchers to achieve reproducible science. Factors can be classified into 4 different categories: complexity, technological change, human error, and concerns over intellectual property rights [24]. Those areas will be discussed in detail below.

1.2.1 Complexity

Studies in some areas require an exceedingly amount of fundamental knowledge. Hardware with certain configurations, specific software packages, and several different programming languages are also needed. Limited access to knowledge, lack of institutional facilities such as labs and equipment, and lack of funding to carry out experiments in the experiment all make research far more difficult to reproduce [24]. To solve the existing problem, detailed annotations and extensive documentation should be used when conducting the experiment. Open software and free access to the prototype implemented can help solve access problems and also make research more accessible to other researchers.

Insufficient amount of time and money for research is also the reason that contributes to the problem. There is the pressure of a tight schedule to the approaching deadline of needing to publish in some influential journals to apply for grants and scholarships for further research [35]. The lack of proper supervision to provide guidance and instructions on the location of stored resources and the proper usage of the equipment further aggravates the problem [35]. One way to solve the crisis would be to properly document open workflows such as the material, procedures, and code used involved in either a physical copy or an electronic copy that are both accessible to the open public [35].

One problem associated with some data access could be the restrictions placed on the data. There could be special constraints placed on the data, and the viewing of the dataset can only be done under supervised civilian arrangement. Data retention policies are another problem. For example, the ADRN Data Rentional and Destruction Policy in the UK limits the research data archival timeframe to a maximum of five years [36].

With an increasing number of abundant useful data, there are many problems that accompany it. Datasets with highly detailed personal information require extra caution on privacy usage, large volumes of data require algorithms with highly computational complexity to run in a sufficient amount of time, and unlabelled raw data require preprocessing of the data so the algorithms can take advantage of the data and apply useful data analysis to it [27].

Figure 1.2 is adapted from the concept of an Open Data Platform and shows the Open Data Platform with named components that make accessing data easier and more efficient [7]. The original concept grouped the requirements of an Open Data platform into five different categories: standardisation, API, integration, policies, and materialisation, so the model provides a high standard for data access. The combination of standardisation and API enables automatic processing and access to the data; materialisation and policies enable stable and smooth data access with provenance; and integration allows different data sets to be shown on one platform [7].

1.2.2 Technological Change

Software and hardware have been updated very rapidly over the last decade. Experiments conducted with an older version of the software can be unreproducible once the software has updated its version or if the software has discontinued its copy. Therefore a code modification is required for the same project to run successfully.

Yale Law School Roundtable on Data and Code Sharing has concluded that "[c]omputation is becoming central to the scientific enterprise, but the prevalence of relaxed attitudes about communicating computational experiments' details and the validation of results is causing a large and growing credibility gap. Generating verifiable knowledge has long been scientific discovery's central goal, yet today it is impossible to verify most of the computational results that scientists present at conferences and in papers" [36].

When dealing with data management at the university, certain problems occur, such as archival time for the preserved data, selective documentation on the subjects related to a few chosen publications, and the lack of effective collaboration tools [36]. Data storage might allocate data to different physical locations due to limitations in data size storage and administrative boundaries [27]. The change in the data repository and the lack of

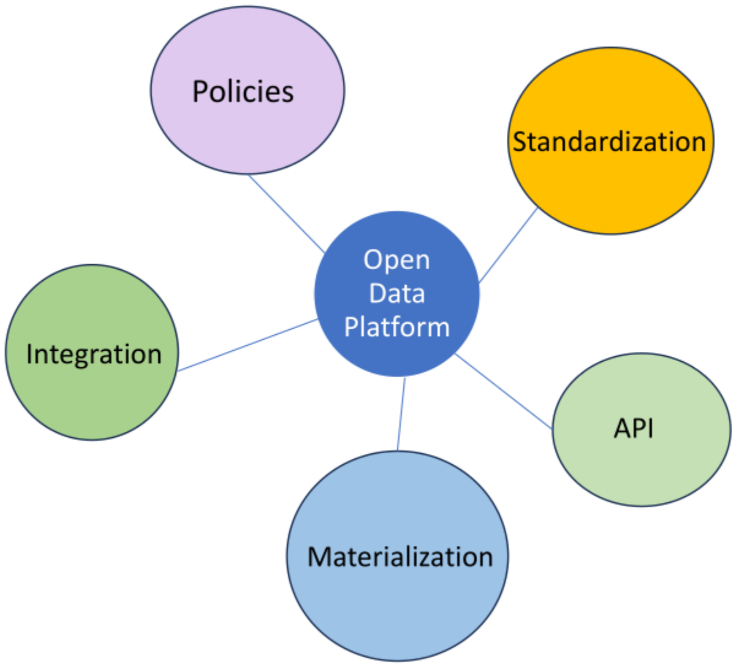


Figure 1.2: Open Data Platform [7]

documentation on posting the new data repository location can cause trouble for researchers who want to reuse the dataset.

1.2.3 Human Error

The researchers sometimes unintentionally forget to elaborate and describe in detail their data collection despite multiple reviews [24]. A detailed planned workflow for researchers to follow will most likely decrease the chances of human error along the way. The planned workflow includes planning, arranging, and recording the scientific analysis: gain access to the data, apply scientific analysis to the data, record the results, and place them in a platform so they are 'reproducible' [36].

Researchers can fail to collect or miss documentation on some data that could be crucial for reproducing. A well-documented research process can help minimise errors and the chances that an unexpected event occurs. Multiple copies of data can be kept to prevent the loss of information when transforming raw data to clean data. The raw data are stored at multiple stages of the experiments at multiple locations for security purposes. The problem with the data is that they can be altered or corrupted during the data cleaning process. The reproducibility of the experiments has increased its difficulties when encountering a large amount of data, especially when the methods are sophisticated and hard to reproduce.

1.2.4 Intellectual Property Rights

Researchers may ponder on the idea of sharing data or code with other researchers with the fear that others will use them incorrectly and irresponsibly [24]. The researchers spend time uploading their data and writing up their detailed documentation is well deserved with recognition and credits in the scientific community. However, the lack of proper citations and credits given to them often decreases the chances of data and code sharing.

Collection and access to personal data often invoke privacy issues, such as who can gain access to those data; without economic value and ownership dispute, which organisation will collect, organise, and archive those online data under strict privacy rules [27]. More often, the majority of the data are collected for profit for specific uses by certain organisations that have no intention to share the data with the public [27].

1.3 Designing PIDs for Time-Series Data

Parts of the paragraphs in this subsection are extracted from a conference paper called "Designing PIDs for Reproducible Science Using Time-Series Data" written by me and my supervisor [37].

A persistent identifier (PID) is a long-lasting reference to a digital resource. The use of PIDs allows specific data to be queried and accessed from a request originating outside the system [37]. One form of persistent identifier used is called Archival Resource Keys (ARKs). ARKs provides a flexible and inexpensive way to make data more accessible [37]. The integrated metadata registry provides the user with one single search that circles through all the different existing individual databases for the data requested. In the paper, the Almanac of Minutely Power dataset (AMPds) is used. The dataset contains electricity, water, and natural gas metering data. Different consumption levels are measured by the sensors at the corresponding timestamp which are measured per minute. The timestamp is used as the primary key when an ARK URL format is used to extract the exact metering data needed.

Non-persistent data can be unintentionally altered or corrupted during the processing stage. Storing the data in the indicated location is helpful and secure for future usage. Persistent identifiers, such as ARKs, can provide long-term access to the data without worrying about losing any information. Time-series data is the accumulation of chronological observations, and the nature of time-series data includes: large in the number of data collection quantities, and the entire dataset is updated continuously [38]. Time-series data are often in the form of key-value pair data [39].

Data that has consistent data structure with descriptive, informational headers are easy to rearrange, model, and extract useful information to visualise and analyse [24]. Putting the data in a single database helps with data analysis. The goal of having the database is to create a repository to store time-series data that is organised using Timeline Index data structure, to manage and improve the stage of analysis [39]. The TimeIndex structure includes the DateTimeIndex object, which often means that the database has a timestamp as its primary key, so efficient query analysis can be performed.

Database and datasets with an ARK implementation allow easy data collection, management, and storage. Persistent identifiers allow users to properly cite them when used in their research, making the exact datasets accessible and open to other users when a reproduction of the experiment is required.

The user inputs a PID URL into the system and the server analyses the URL and fetches the data from the storage. The data is formatted in the process and sent back to the user. In the process of retrieving the data, there could be different storage locations from the servers. The different datasets exist on different and separate servers on the web. The use of an HTTP server can improve the client service and strengthen the protection and security compared to other databases [39].

1.4 Our Contributions

In this thesis, the following contributions were made:

1. Contribution to the IEEE Standards Association P2957 Data Governance and Metadata Management Working Group. We created detailed definitions for all components in Figure 17 of the proposed Draft; a block diagram called the Federated Metadata Registry, which is a proposed data persistent architecture (see Chapter 2).

2. Evaluate existing PID standards and protocols with the goal of reproducible and repeatable science. Create an initial design that allows data to be downloaded with a single-click in the exact way used by the authors in their published experiments. By single-click we mean with one click of a URL the system automates a number of manual tasks (see Chapter 3).

3. Build a proof-of-concept prototype that demonstrates how PIDs can be used with time-series data given our initial design in (2). In addition, demonstrates how specific data extraction can be done without the need to write code (see Chapter 4).

To illustrate these contribtions, imagine the user just read a research paper and wants to reproduce their results. The paper mentions that they used 10-fold cross-validation for testing a dishwasher model (DWE) to track voltage stability (columns TS, V). The dataset (AMPds) can be downloaded using a DOI link. What does the user need to do to get the data ready to use?

The user would have to:

1. Click on the DOI link to go to where the dataset is published

2. Download the dataset (AMPds.zip)
3. Find the relevant file that contains the data used (DWE.csv)
4. Write code to isolate the timestamp and voltage columns (TS, V)
5. Write code to hopefully split the data into 10 chunks exactly like the authors did

OR the user could just click on this link to do all the above 5 manual steps:
https://n2t.net/ark:/57460/AMPds/DWE/V@*/10

1.5 Thesis Organisation

Chapter 1 is the Introduction chapter that introduces reproducible science. It also provides some common difficulties in finding data for reproducibility, including complexity, technological change, human error, and intellectual property rights. This thesis will focus on designing PIDs for time-series data. Chapter 2 covers a block diagram of a Federated Metadata Registry. The diagram includes two registry types: Federated Catalog Registry and Federated Data Types Registry. The block diagram demonstrates how the data can be stored and categorised for ease of further processing of the PID system and accessing the data. Chapter 3 Data Access Architecture discusses different types of Persistent Identifiers. The prototype is built based on the Internet, the Hypertext Transfer Protocol (HTTP), and the Representational State Transfer (REST) software architectural style. Chapter 4 will further explore the concept of using ARK the persistent identifier for our implementation and demonstration. Detailed PID design and prototype design are included in this chapter, as well as some lessons learnt. Chapter 5 includes the Conclusions and some future work regarding the proprietary implementation.

Chapter 2

Data Persistence Architecture

It is important to have a standard reference architecture that can cooperate with diversified types of datasets from different repositories regardless of their data source and structure [29]. Part of the investigative work done by IEEE Standards Association P2957 Data Governance and Metadata Management Working Group (BDGMMWG) is to look at how to create a data persistent and discovery system that points to specific data from a given data repository. BDGMM desires to create a standard data infrastructure that improves the ability to locate and access digital data across heterogeneous repositories under the FAIR (Findability, Accessibility, Interoperability, and Reusability) data principle without worrying about the data source and data structure [29]. The IEEE Standards Association BDGMMWG Working Group defines a framework for implementing metadata and big data governance [40]. The architecture requires additional properties of the persistent identifier, such as using it to store the location of the digital object and to state the core metadata information on top of the basic function as a long-lasting reference [29]. The use of persistent identifiers in combination with the structure of the data management framework shown in Figure 2.1 aids in data discovery by composing the data into machine-actionable structures using machine-readable formats through standard endpoint services [40]. The structure of the metadata management framework is incorporated into a figure, referred to as the diagram in Figure 2.1 Block diagram of a Federated Metadata Registy. Figure 2.1 illustrates a proposed data persistence architecture, factored in the proposed guiding metadata management framework design criteria and created by the IEEE Standards Association BDGMMWG Working Group, which shows the two federated registries in which the system can map the data [8]. The data management framework regulates the integration of data among diversified domain repositories [29].

The potential BDGMM Reference Architecture, also known as Figure 17 from BDGMMWG, shown in this thesis as Figure 2.1 summarises three key operations: discover, map, and access. The three operations are feature components identified and classified from The Big Data Governance and Metadata Management: Standards Roadmap [29]. The three operations are explained with comprehensive illustrations that can be used as guidelines for anyone reading the IEEE whitepaper. The Discover option allows users to retrieve datasets from the Federated Catalog Registry by searching through the data model, schema, definition and metadata at their core, domain, and relationship with other datasets [29]. The Map operation allows users to find and retrieve other datasets that share similar data models from the Federated Data Types Registry according to their types and properties [29]. The Access operation allows the system to retrieve one or more available datasets selected from the previous operation stage [29].

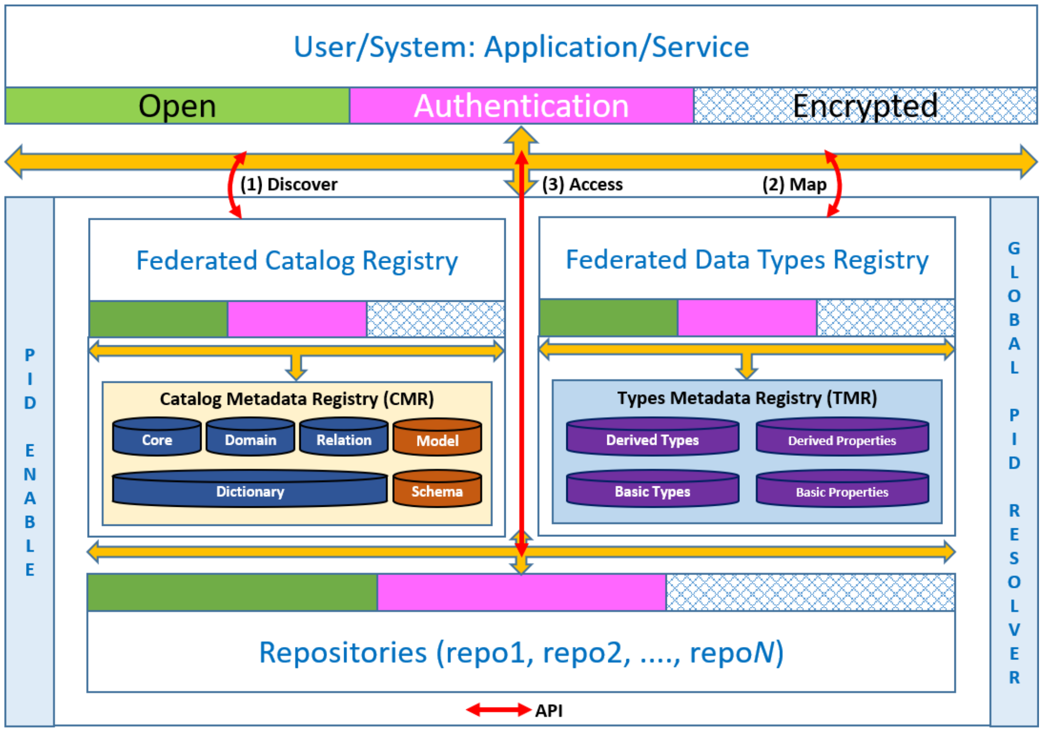

User/System: Application/Service
Open
Authentication
Encrypted
(1) Discover
(3) Access
(2) Map
P I D E N A B L E
G L O B A L P I D R E S O L V E R
Federated Catalog Registry
Catalog Metadata Registry (CMR)
Core
Domain
Relation
Model
Dictionary
Schema
Federated Data Types Registry
Types Metadata Registry (TMR)
Derived Types
Derived Properties
Basic Types
Basic Properties
Repositories (repo1, repo2, ..., repoN)
API

Figure 2.1: Block diagram of a Federated Metadata Registry (BDGMMWG Draft Fig. 17 [8])

According to two researchers at the University of Berkeley Meyer and Lüling stated that in 2003, there would be roughly 1 exabyte (= 1 million terabytes) of data generated every year; and estimates of the entire worldwide data would double every 20 months [41]. With a large amount of available data, fast and accurate automated processing of massive amounts of data across different domains requires implicit details about their data types.

Digital Object architecture contains components such as an identifier registration and resolution system, as well as a storage location such as a repository that provides access to the data [29].

Details about the data can be used as the keyword for classification and they can include the form the data is in and any unique instances of datasets associated with them. The metadata specifications provide descriptions of data objects and their type [42]. The concise and clear terms used and the controlled vocabularies enable computers to accurately locate the information and extract processing data for users [42].

Data can be grouped and classified by common patterns and core information when the data is broken down into specific structures [41]. To access the data with one single search, the data federation combines all the siloed data stored to form one extensive database. Most of the data are difficult to discover and preserve because they are found in many small datasets held by numerous different independent researchers across different disciplines and facilities [42]. Data federation stores data in a heterogeneous set of data siloes. The extraction of data from multiple datasets has many challenges. The challenges arise because the data have different dimensionalities and formats compared to other data that are also going to be integrated into the unified data federation, as well as their complexity, the quality of the relevant data excluding noise, and similarities such as mutual concordance among different data [43].

The integration of data federation involves combining multiple database systems that are independent and physically distributed over multiple heterogeneous data sources to form a unified view for users [44]. The user who uses data federation should be able to access different types of database servers and files with various formats. It should be able to integrate data from all those independent and separate data sources. The data should be accessible through a common set of APIs.

2.1 Federated Catalog Registry

For researchers conducting multidisciplinary research, the need to search through multiple catalogs to find the appropriate data from each catalog is very time-consuming, tiresome, and prone to human error. The struggle of working with different catalogs is that different organisations construct their catalog system with different schemas and the data is stored in very different formats [45].

Data scientists must work with high-quality and timely information that is easily accessible and integrative to achieve desirable results [46]. Most of the time, data scientists spend too much time looking for the availability of and ways to access the data. Although there were relevant datasets, there are many barriers that prevent users from accessing them. Such barriers include: governance-related barrier (laws and regulations regarding sensitive data), cost barrier (a cost is associated with the access of the data) and time barrier (a time delay of the data open to public) [47].

Even after the user found the useful information, working with multiple datasets that have different data models and terminology systems is difficult to operate, and thus the data within different systems are not interoperable [48]. Too much time is spent searching for the data on different datasets platforms instead of trying to understand them. Working with inadequate datasets with inconsistencies in their data quality would result in incompetent research analysis results.

With a Federated Catalog Registry, a global schema would be shown to end users. The complicated multiplicity of distinct catalogs would be hidden behind the interface, leaving one simplified and unified catalogue [49]. The Federated Catalog presents a standardised data query interface and metadata model exposed through an Application Programming Interface (API) [49]. An API is defined as “the interface to a reusable software entity used by multiple clients outside the developing organisation and that can be distributed separately from the environment code” [50].

Federation means that a digital identity is referred to identically across all platforms even when changes are applied to the identity [51]. Catalog service provides the ability to search for the description of the data and related resources and supports the user for publishing [52]. A Federated Data Catalog will provide a single unified interface / API of all data collected despite the location of the origin if the user needs to search the platform [51].

The single point of access API allows data to be indexed, searched, rearranged, and extracted [53]. The registry is a collection of the available distributed information and puts them in a unified view, as well as enables support and tools for notes and comments to the resources if needed to achieve a high standard throughout [54].

Catalog Services are software components that “simplify data discovery, facilitate data transfer, and help [with] data fusion”; they help the user discover and locate the right data [45].

The use of the registry helps researchers to find, appreciate, and correlate the needed information instead of searching through pages and pages on the Internet [54]. Researchers who made and published their datasets on the registry would also benefit from within; there will be increased exposure, usage, and recognition for their work, as well as complimentary suggestions and recommendations for the API platform [54].

The Federation Registry stores the core registry information. The Registry comes from the Latin word registrum, which has the meaning of a list or catalog [55]. The characters and structural description of the metadata are incorporated in the design of the Federated Catalog Registry.

2.1.1 Catalog Metadata Registry (CMR)

Metadata is described as data describing other data. Under some circumstances, the user can be unaware that there is available data in the relevant field they are looking for; or do not have access to the data that some organisations possess, or do not have the right software tools to perform the calculation after the retrieval [56]. Therefore, metadata must provide precise and clearly defined applicable information about the data it describes [57] for easier search and retrieval. Metadata is kept, organised, managed and accessed in a database called a registry or repository [58]. Metadata registries organise information about the data, grant permission for data usage, and help to share data [58]; it keeps information about the characteristics of the data that helps the operation of data sharing and exchange in different disciplines [59]. Metadata in catalogs gives the characteristics of the object for further processing, querying, and analysis [60].

The best way to utilise metadata is to normalise and convert the data into schemas, summarise, and assemble them into usable information for the API [61]. To function well with API, catalog data that were once well understood by humans need to be understood and processed by machines without the intervention of people [61].

The Metadata Registry provides the description and entities of the objects. Different metadata view data from different perspectives. Different sets of metadata are considered compatible if their entities overlap. Metadata interoperability can also be interpreted as a degree of measure of compatibility between different metadata sets [57]. This registry provides two basic purposes "end-user oriented purpose" and "production oriented purpose": to support potential users and to support the duration of the process such as preparation, operation, and assessment [58].

The Data Catalog behaves like a data storage for metadata and serves with some data management tools for data analysis. Datasets are files and tables that data scientists work to find and access. Not only does a data catalog operate like a data inventory, it also functions well for search, access, and computation.

Metadata can include the data source, origin, owner, and other attributes of a dataset as well as domain-specific attributes, so in the example of the metadata used for climate datasets will be different compared to the metadata for power or for energy. A catalog metadata registry creates a central repository for the data and the metadata to ensure consistency and accuracy. Metadata gives context around data and helps document data so it can be shared and reused across multiple use cases.

With a growing number of open data available, researchers often have to request approval of access while searching for the data in the metadata registry and wait for the response afterward. Data accessibility includes the following forms: open access, open access with barriers that can include broken links, and regulated access that requires approvals or payments [47]. In the area of searching for digital health information, particularly in the area containing ophthalmological health information via MEDLINE, Google's search engine, and Google Dataset Search, 94 datasets are open access, 27 datasets are open access with barriers, and 19 datasets are regulated accessible, totalling 140 unique datasets [47]. Of the 94 datasets, which accounts for 67 percent of the total ophthalmological datasets found, there is a 2-week grace period for the database that requires an email response to access open-access datasets [47]. The time the user spends waiting can alternatively be allocated to understanding and validating the data.

Catalog Metadata Registry (CMR) enhances data management and accelerates data-driven decision making by providing an effective unified platform to host all the data required for making a prone decision. The verified unified data catalog registry provides quality validated datasets which gives the user trust and confidence in using the registry.

Instead of searching through multiple registry, searching through one shortens the search time by a significant amount. With a Federated Catalog Registry, all the separated catalog registries are compiled into one single access so that the user can search and look for data more efficiently, making quicker and more informed choices on data selection. In the example of medical data, the ease and fast access to researchers is an important factor, the use of a Federated Catalog Registry prevents users from worrying about the complications of licencing agreements or ethical committee approvals [47].

A standard metadata model "provides a methodology for the registration of the descriptions of data elements" and the description of the data element is represented through its components [48]. In Figure 2.1, Catalog Metadata Registry is the standard metadata model and each unique different data element is represented through the components: core, domain, relation, model, dictionary, schema. In the Catalog Metadata Registry, there are 6 types of metadata: core, domain, relation, model, dictionary, and schema. Each is explained in detail below.

Core is the main classifications and groups among the Catalog Metadata Registry. Core is the metadata that are common in all datasets; e.g., dataset name, size, type of data, to name a few. Metadata describe a dataset and ensure that the data are FAIR: "Findable, Accessible, Interoperable, and Reusable" [62]. Metadata allows users to sort and detect specific structured data such as documents written in text or unstructured data such as audio, images, videos, web pages, *etc* [63]. Metadata is stored in a database and generally manages unstructured data by providing a common framework and uses examples such as meta tags that include description and keywords to identify, classify and describe the content within a web page [63].

Domain is the metadata for a particular field or area. For example, medical data, power/energy data. Domain can be a category or a group of data elements that share the same purpose or meaning. Within a specific database, such as a customer database, some common domains include the customer name, address, email, and phone number [64]. Data domains provide an accurate and consistent way to govern and maintain data quality by providing a framework to ensure that different sets of rules, constraints, and standards are used to govern the format, length, and type of data that can be stored within each unique data domain [64]. Domains can be classified into three categories: environmental domain, object class domain, and object format domain. *Environmental domain* is "the discipline and the community that the [metadata] services" [65]. A database such as PubMed is a database of citations used in the fields of medical research, and the Canadian Centre for Energy Information is used in the government sector for the sum of energy consumption. *Object class domain* is "[a]ssembly or grouping of similar objects by type" [65]. For example, different activities can be classified as unique domains according to their places or events. *Object format domain* is "[the] object's composition and what it is made of" [65]. For example, the Content Standard for Digital Geospatial Metadata (CSDGM) and the Federal Geographic Data Committee (FGDC) contain digital geospatial resources; and Dublin Core contains digital resources such as websites and physical resources such as CDs and books [65].

Relation is “information about another resource that is related to the current resource” [66]. The relation can be entered into the metadata form in two formats: dropdown menu or text entry; in both cases, the titles of the objects, the identifiers of the objects, and the permalink to the record of the objects need to be included [66]. The direction of the relationship should specify which object is created first and the word choice for describing the relation should be chosen so that it represents and references the other [66]. Relation links the objects from a single item to a more sophisticated and meaningful linkage for defining a more purposeful and relevant concept [58].

Dictionary includes framework and construction of terms which includes the composition of elements of the terms and the mechanism on how to form them [58]. The defined vocabularies, definitions and terms aim to eliminate ambiguity and provide a more defined relationship between the terms for validation [67]. The dictionary can link conceptually similar objects from distinctly different schemas despite coming from different sources and having different models or mappings [68]. Dictionary can be a storage place for controlled vocabularies. A controlled vocabulary contains item lists such as “headings, thesauri, ontologies, and taxonomies” to make the data more findable and accessible by providing a consistent way to construct the data [69].

Model provides a user-friendly representation and follows an iterative process that prepares the metadata to explore and publish [70]. Models can describe physical raw data files that are in the form of binary, images, or text containing numeric values[71]. IBM plans the workflow of the metadata model that begins with the collection of structured data sources information to meet business requirements [72]. The University Library at UC Santa Cruz describes the process of building a metadata model as follows [73]. When building a model, take into consideration what form the object is in (e.g. a physical object, a digital object, a digital representation of a physical object, etc.) and the necessary information needed to describe the object. Then, the essential information required to facilitate the discovery of the object and provide sufficient context for identification is listed. Using the list built from above with all the information listed out, build a schema or use an existing schema such as Dublin Core and store the information into the schema.

An example of the metadata model is the ORACLE-created ATG Customer Intelligence Data Warehouse, and details about the metadata model will be discussed in the following [74]. This data warehouse is a relational database for performing enhanced analysis for business users and report creators. There are three layers to the model: database view, model view, and dimensional view. The database view shows lines linking the query subjects showing the relationship and cardinality between them. The model view combines the Data Source Query Subjects into more informative information for the user. The Dimensional view is a more abstract form of the report of the data, specifically targeted towards the users.

Schema is a set of semantic properties used precisely to best describe corresponding metadata elements [67]. Schema is also called Formats or Element Sets and examples of schema include MARC 21, Dublin Core, MODS, and ONIX [67]. Existing and predefined data elements and properties are used rather than building them from the ground up [61]. Schema can also be defined as an entity that defines elements or properties [75] or an entity that represents a set of arbitrary but specific and independent data elements [57]. A metadata schema is a list of elements that are used to capture information about a data element [76].

An appropriate metadata schema should be selected to best fit the discipline or repository for which the dataset is used. The metadata schema would have an overview structure and would have the appropriate metadata information that can be applied to the data element. Depending on the chosen schema, different standards will be applied. Dublin Core is a generic schema that is widely applied and can be extended with enhanced additional vocabularies for more specific information [77]. Dublin Core, also known as Dublin Core Metadata Element Set, contains 15 term names to describe digital resources such as images, video, web pages, sound, etc. and physical resources [78]. The 15 elements of Dublin Core are: contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title, and type [79]. The Dublin Core Metadata Element also includes some additional vocabularies for greater coverage and more specificity in the defined areas [79]. If a generic schema is not the right fit for the data elements, a more specialised domain-specific schema can be used instead.

The domain-specific schema usually has a more defined structure and more area-focused vocabularies that are only applicable to the specific domain understood by specialised researchers in that field [77]. Metadata needs to be standardised to be effective, and that includes standardising language, spelling, data format, etc. [77]. Part 3 of the ISO11179 standard organises metadata elements into five categories: identity, definition, relation, representation, and administrative [57]. The complete set of all attributes of the data element described from each of the categories represents the definition of that specific schema [57]. The W3C Resource Description Framework Schema (RDFS) is another example of an existing schema and provides a common data model that is currently working with the Web Ontology Group to increase the specificity of schema vocabularies to reach a broader audience [80].

Standards can help users generate the term names to describe digital or physical resources within a schema. There are three types of standards: content standards, data value standards, and data structure standards [73]. Content standards describe where the elements come from and what would be the best use of the elements. Data value standards are lists of standardised terms, such as controlled vocabularies, subject terms, genre terms, etc. Data structure standards encode metadata information to ensure that it is machine readable and can be understood by humans [73].

2.2 Federated Data Types Registry

Data Types are recognised and registered characterisations of data [9]. Registered data types are used to illustrate data by humans, as well as to assess and process data to make it machine readable [9]. In the paper RDA Data Type Registries Working Group Output, a proposed workflow for the registered data types illustrated in Figure 2.2 is shown as follows [9]. In Step 1, the workflow starts when the user encounters an unknown type of data that is shown in the format of id, type, and value triple. Once the item with the format such as an identifier resolution system or a data repository is received and detected, the registers can analyse the type and structure for further processing. In Step 2, the user would query the type registries with the unknown types. In Step 3, Type Registries send responses such as data type, relationships, and properties according to suggestions to external resources that can advance to the next step. In Step 4, the typed data would be sent to type-appropriate services for further analysis for details.

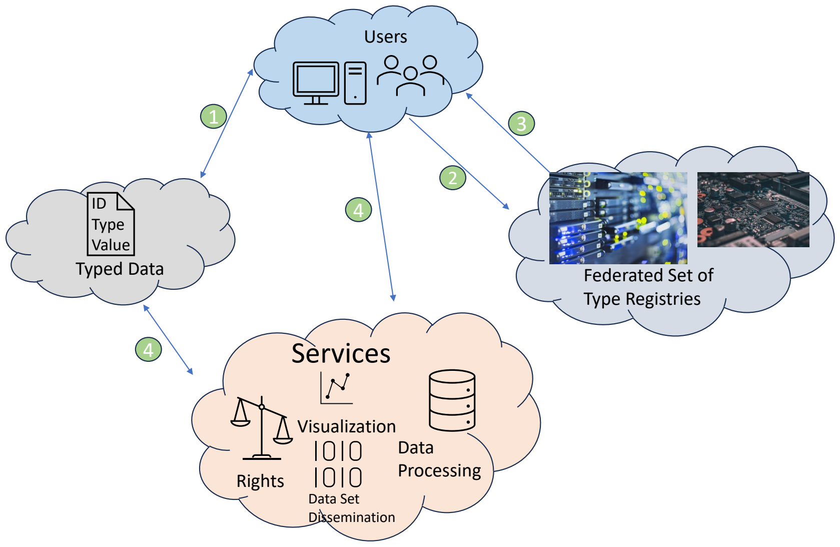


Figure 2.2: Workflow of the Data Types Registries [9]

The metadata registry contains the "definition of structure and semantics of data, [and not holds] the actual data" [81]. Every registry entry has a data type that the data entry can store [82]. Data types are vocabularies that can be organised and related to each other and a registry stores and provides access to metadata [83].

Windows registry is a complex relational database that stores a large amount of information with strictly classified data types [84]. Data types specify the kind of data that can

Table 2.1: Registry Data Types [1, 2]

Constant	Description
`REG_BINARY`	Binary data in any form
`REG_DWORD`	32-bit number
`REG_QWORD`	64-bit number
`REG_DWORD_LITTLE_ENDIAN`	32-bit number in little-endian format, equivalent to `REG_DWORD`. In little-endian format, the value number is stored from the lowest byte to the highest byte. Eg.0x12345678 is stored as (0x78 0x56 0x34 0x12)
`REG_QWORD_LITTLE_ENDIAN`	64-bit number stored in little-endian format, equivalent to `REG_QWORD`
`REG_DWORD_BIG_ENDIAN`	32-bit number in big-endian format. In big-endian format, the value is stored in memory from the highest byte to the lowest byte. Eg, the value 0x12345678 is stored as (0x12 0x34 0x56 0x78)
`REG_EXPAND_SZ`	Null-terminated string that contains unexpanded references to environment variables, in either Unicode or ANSI string Eg. "%PATH%"
`REG_LINK`	Null-terminated Unicode string that contains target path of a symbolic link
`REG_MULTI_SZ`	A sequence of null-terminated strings that are terminated by null characters (\0). Eg. String1\0String2\0String3\0LastString\0\0 The first \0 terminates "String1" and the second-from-last \0 terminates "LastString" and the final \0 terminates the sequence.
`REG_NONE`	No defined value type
`REG_RESOURCE_LIST`	Device-driver resource list
`REG_SZ`	Null-terminated string. It will be a Unicode or ANSI string, depending on whether the function is in Unicode or in ANSI functions.

be stored in the database; each stored value contains the three components in this arranged order: name, type of data, and value of the data [84]. In Table 2.1, specific registry type is illustrated by the described data types and their definitions are given by the document written by Microsoft for Windows App Development [1, 2].

2.2.1 Types Metadata Registry (TMR)

A metadata registry is a database that is used to store, organise, manage, and share metadata [58]. An issue often encountered in building an exhaustive metadata registry is that participants may have used diverse schemas and description methods to create their metadata records [85]. Building one integrated metadata registry simplifies and enables one search for the end-user to retrieve the information needed rather than searching multiple

Table 2.2: Derived Types

Basic Types	Corresponding Derived Types
text	character, varchar
number	integer, long, real, float, double, percentage, scientific
currency	USD, RMB, $, Euro, Yen
boolean	check box, yes/no, true/false, on/off
date/time	timestamp, short date, medium date, long date, time am/pm, medium time, time 24 hour
blob	rich text, attachment, memo, attachment
calculated	lambda function, imaginary number
pointer	hyperlink, lookup

Table 2.3: Basic Properties

Basic Properties
Centre of data
Skewness of data
Spread among the data members
Presence of outliers
Correlation among the data
Type of probability distribution that the data follow

times through all the different existing individual databases. Therefore, interoperability will be a critical component in such development. Most of the time, different types of resources are already classified under a variety of specialised schemas [85].

Data Types are characterisations of data at any level of granularity. Data types identified, defined, and registered data allowing for different operations on similar datatypes. For example, numeric datatypes can have mathematical operations applied to them; however, text datatypes cannot. When encountering unknown data, the registry can explicate those types and output the type definitions, relationships, and properties. *Basic Types* are used to interpret and later process data. A Data Type Registry provides a detailed and structured description of the inputted data. Some of the basic types are listed below in Table 2.2 (left column). *Derived Types* are more specific cases/instances of the basic types. For example, the derived types are listed in Table 2.2 (right column).

Basic Properties describe the nature of the data. These descriptions are essential to understanding and performing any statistical analysis on the data. Different Exploratory Data Analysis (EDA techniques) can be used to identify the properties of data so that appropriate statistical methods can be applied to the data. Here is a list of different basic properties of a set of data as shown in Table 2.3. *Derived Properties* are properties that do not exist on the Entity Type associated with the Entity Set; rather, it exists on a type that is derived from the base type of the entity set.

Chapter 3

Data Access Architecture

3.1 Persistent Identifier (PIDs)

PIDs accelerate the adoption of Open Science by supporting reproducibility and building provenance for citations [86]. Users can locate and find the resources they have used for their work by tracking and following the PIDs. The presence of a persistent identifier assists and benefits in many areas: aids in citation tracking, aids in reference, and gives the right authors credit for their work [87].

The Internet is a data communication platform that allows for finding and locating digital objects with ease. However, as time goes on, the address of the digital object may change and the resource it contains may become unavailable. To solve this ongoing problem, the usage of *persistent identifiers* (PIDs) allows and maintains the persistent record of digital objects.

Persistent identifiers can efficiently and reliably retrieve the cited work when most data is stored in highly decentralised locations across many online databases [88]. URL is a common way to create a link to the resource. However, re-configuration on the website could cause the link to break because the URL identifies the location and not directly the content of the resources [86].

This process of URL not being found is known as *link rot* which is common nowadays when the availability of the cited works decays with time [88]. Figure 3.1 is an example of URL link rot. Link rot can be caused by many factors, including human errors and technological failures. Technological failures such as server failure; and human errors such as reorganisation of the website and transfer of content from one publisher to another without updating the old website can all cause the link to break and the content to be lost [89].

In the field of academia, when URL changes, the corresponding existing reference would break and this *reference rot* impacts 1 in 5 publications [88]. The consequence can be more detrimental if the linked resource is crucial in legal, medical or scientific areas [89]. Peer

Not Found

The requested URL was not found on this server.

Figure 3.1: URL not Found

reviews or any future work that relies on the digital resources used before will become untraceable after the incidence of reference rot.

The use of persistent identifiers allows continued access to the resource even when the original location changes, which prevents the occurrence of the impermanent nature of the URL and enhances data management [90, 89]. PIDs are implemented in research workflows to enable a trusted digital network between objects, contributors, and organisations, which alternatively increases interoperability between resources [90, 89].

A Persistent Identifier (PID) for a digital object is in the form of a permalink. It can be thought of as a permanent reference to a document, a file, a digital object, a physical object, a contributor, an organisation, etc. [91, 90]. PID identifies information unambiguously and offers a reliable way to track citations and metadata usage across different platforms [92].

A persistent identifier is a unique identifier pointing to an entity without ambiguity and is often represented as a string of letters and numbers [93, 94]. In the event that different digital objects have name changes or share the same names with each other, PID eliminates confusion and always points to the desired digital object in a persistent way. Organisations actively maintain and organise short and descriptive URLs that are called permalinks, so that they are updated and always point to the right metadata [91]. The consistent permalink restricts and limits the possibility of URL changes to avoid the entity becoming unlocatable [91].

When the research is cited correctly in the case of scientific research, credit will be given to the rights holder. This will promote more reliable citations in the community, and recognition and trustworthiness will accumulate. When there are updates on the versions of the published work, each version of the item is identified by a new persistent identifier that maintains a provenance trail for each item and all its versions [86].

The issue of a persistent identifier for an entity ensures the long-term availability of the item [86]. With the presence of PIDs, digital objects are becoming more: discoverable, accessible, useable, intelligible, interoperable, and assessable [93].

The wide use of Persistent Identifiers promotes linking and interoperability on different platforms and has many other advantages, including ease of locating the resource and clear distinction of different versions of a resource if there is a metadata update or correction [86].

The process of using persistent identifiers enhances the ease of finding between researchers and makes their research work and results more accessible to others [95]. This process further advances the use of open science and makes the recognition and assignment of metadata to academic publications more accessible and straightforward [96].

Persistent identifiers are an important infrastructure for the long-lasting preservation and maintenance of data that reduces the occurrence of link rots and could be a solution dedicated to web archives [89].

The flexible, coherent and smooth integration of PID services in all disciplines and throughout different stages of the research lifecycle is the key to populating PID usage among people [86]. Machine-readable PIDs are essential resources for granting and accelerating information sharing across systems [90]. Incorporating persistent identifiers in published research work and online journals will encourage and consolidate a unified citation style guideline [87]. Furthermore, the ability to use a persistent identifier to track the number of usages for citations, discussions, and sharing of an article could help the research community measure the impacts made by each individual author [87].

However PID also has its limitations and weaknesses. PID still relies on humans to monitor them. In addition, many PIDs still require money and budget from the organisations for creation and maintenance. The long waiting process for PID creation would also be one of the limitations of using them. Therefore, to minimise the existing weaknesses and limitations, picking a PID that requires a low creation time and is free to use would be preferred among other options.

There are many different persistent web-based identifiers with different requirements and each of them targets different digital entities, physical entities, living entities, and descriptive entities [88]. Users must consider the scope and lifecycle of the data before choosing among all existing identifiers to optimise data extraction [88]. To achieve persistency for entities of different kinds, multiple types of PID can be used depending on the stage of development, predetermined by requirements and the length of the time commitment [86]. In the following paragraphs, different examples of PID will be introduced: Open Reseachers and Contributor Identifier (ORCiD), Digital Object Identifier (DOI), Research Organisation Registry (ROR), Archival Resources Key (ARK), International Standard Name Identifier (ISNI), Handle (HNDL), Persistent Uniform Resource Locator (PURL), Universal Resource Name (URN) and some Organization Identifiers.

3.1.1 Open Researcher and Contributor Identifier (ORCiD)

There is an astonishing growth of more than 15 million ORCiD IDs issued for researchers accumulated from the start of the launch in 2010 to the date of October 2022 [97]. The adoption of ORCiD IDs is becoming popular among researchers. The aim of ORCiD is to provide a persistent identifier that unambiguously distinguishes the work of the contributor as the author, manages a research profile for the researcher [98].

ORCiD is an open registry that connects users with their contributions. ORCiD IDs are created, edited and owned by individual users [99, 86]. An ORCiD ID is a digital identifier that users can associate with their professional work persistently despite name changes or different naming convention formats [90]. ORCiD ID focusses mainly on researchers. It contains links to and information about articles and datasets associated with the researcher [86].

ORCiD works interconnected with other persistent identifier networks. The user with an ORCiD ID can associate and connect his ORCiD ID to existing DataCite DOIs using DataCite's Search and Link service [86]. Through ORCiD's autoupdate feature, if a publication or dataset is published with a new DOI and the metadata contains an ORCiD ID, the authors with the ORCiD ID will be automatically updated [86].

The limitation of ORCiD is that its main focus is to link researchers with their published work. The persistent identifier on which my thesis project is based is on the metadata and storage of the actual data.

Figure 3.2 shows an ORCID iD expressed as a full https URL [10]. ORCID iDs are assigned randomly between 0000-0001-5000-0007 and 0000-0003-5000-0001, or between 0009-0000-0000-0000 and 0009-0010-0000-0000 where every four integers are separated by a hyphen for easy readability.

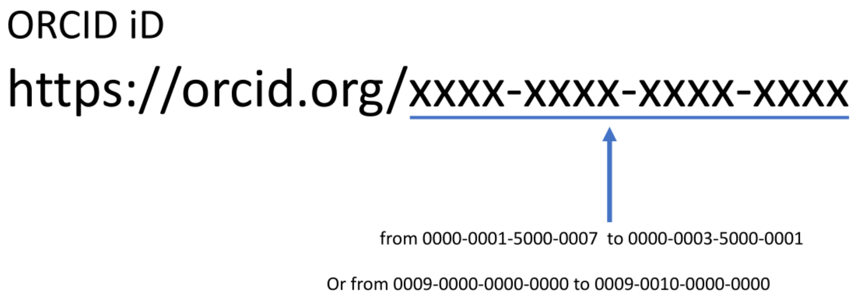

ORCID iD
https://orcid.org/xxxx-xxxx-xxxx-xxxx
from 0000-0001-5000-0007 to 0000-0003-5000-0001
Or from 0009-0000-0000-0000 to 0009-0010-0000-0000

Figure 3.2: ORCID iD[10]

3.1.2 Digitial Object Identifier (DOI)

DOIs are persistent identifiers used for entities in technical and scholarly research publications, such as articles, books, conference proceedings, and datasets [99, 100]. It was initially developed in 1998, and the first applications were launched in 2000 [101]. Until December 20, 2021, approximately 275 million DOI names were assigned [101]. The DOI System is created and maintained by the International DOI Foundation (IDF) and adopted as the International Standard ISO 26324 in 2012 [94, 87] and provides continued access to the metadata for the entity [90]. DOI promotes frequent data sharing and reuse of information by making entities discoverable over a long period of time [87].

DataCite and CrossRef are the two main IDF organisations that manage the registration of a new entity to ensure that no two entities are assigned the same DOI [94]. DOI is issued at the time of the publication of the entity and is represented as unique combinations of numbers that will remain unchanged [87, 99].

DOI number is composed of a prefix, a forward slash for separation, and a suffix: the prefix always starts with the number 10, followed by a dot and a unique four- or more-digit number that is specifically assigned to individual organisations; and the suffix is assigned by the publishers to identify the entity [87]. A DOI can appear as either an alphanumeric string of digits and letters or as a webpage URL. A detailed explanation of the URL is shown in Figure 3.3 and shows an example of a DOI [11, 12]. DOI always follows by a 10 which acts as a general identifier to tell the user that what follows after is a DOI and not just any identifier with digits. The prefix in Figure 3.3 is the combination of 10. followed by the four digits. The slash separates the suffix that follows. Registering Agencies use different combinations of items, such as volume numbers, page numbers, dates, or names, to specify the suffix of the DOI [11].

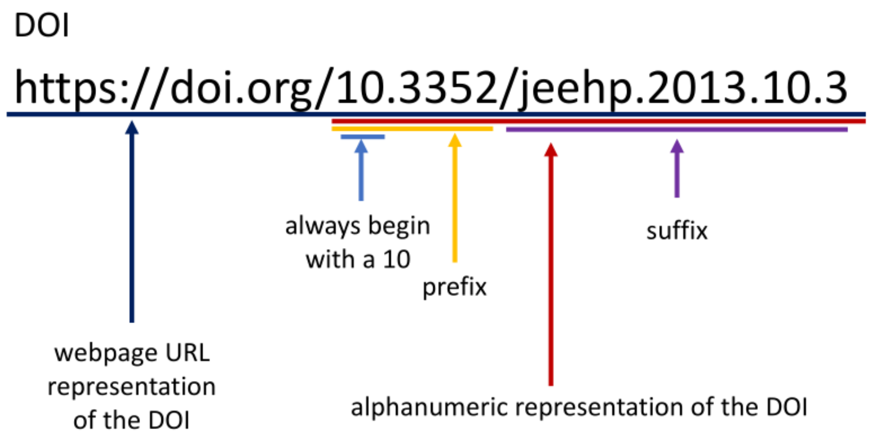

DOI
https://doi.org/10.3352/jeehp.2013.10.3
always begin with a 10
prefix
suffix
webpage URL representation of the DOI
alphanumeric representation of the DOI

Figure 3.3: DOI [11, 12]

Most published scholarly articles are assigned a DOI that allows the resources to be cited, reused, and shared with ease. DataCite supports the creation and management of DOIs by improving the procedure and integration so that the cited entities are openly available, easy to find, and can be utilized well [102].

There are multiple ways to find out if a publication has a DOI associated with it or not. CrossRef and DataCite are the two main International DOI Foundation (IDF) Registration Agencies (RAs) organisations, and they were introduced and previously discussed. Depending on what information the user has and what the user is looking for, a refined search can be conducted on CrossRef or on DataCite. CrossRef provides the option to search the metadata of the publication by title, author, DOI, ORCID, etc; and DataCite has the option to search by specific categories such as: works, people, data centres, members, etc. [87].

DOI is a leading PID in the area of scholarly publications such as journals, proceedings, books, preprints, etc. [86]. The DOI is used mainly to identify documents and is "registry based"; all other metadata related to the DOI are stored separately in the registry[11]. The user can obtain the current location of the document by sending a query to the registry[11]. However, there are costs associated with the establishment of a new DOI. The detailed cost of registering new DOI names depends on the services required and purchased.

3.1.3 Research Organisation Registry (ROR)

Research Organization Registry (ROR) aims to discover, associate and establish a formal connection between organisations and their corresponding research output, so ROR can also be referred to as a registry of research organizations [103]. ROR registry offers open-source code. It provides each unique individual organisation with a distinctive ID under the Creative Common CC0 1.0 Universal Public Domain Dedication; the ID is expressed in the form of a URL, alongside also includes some additional metadata about the organization [103]. ROR registries are free, easily accessible without additional access requirements, and can be found using the following methods: Web, REST API, and datasets in JSON or CSV formats [103]. If an organisation is not found on ROR and the researchers list the organization in existing publications, then any user can request to add the organization to ROR. Furthermore, any user can suggest additional updates to the registry.

The Research Organisation Registry (ROR) contains the ID and metadata for more than 102,000 organisations and can be searched through the following platforms: a search interface, REST API, and data dump [103]. ROR ID database is based on an initial set of data (seed data) from the Global Research Identifier Database (GRID) and stores metadata about organizations [99]. The registry data is processed and examined by the ROR community before being published to the public on a rolling basis [103].

The process of creating an ROR registry and updating it is summarised on the official website and is explained and elaborated in detail below [103]. New requests for registries are submitted and can be found in the issue queue of change requests or the ROR Updates tracker. ROR code is openly available on GitHub under a MIT license [104]. The time it takes from new submission to updates on the website usually takes no more than 6 weeks. ROR registries are created, maintained, and contained in a centralised community-based platform. Once the document submission is reviewed and approved by the ROR's metadata curation lead and advisory board to make sure that the content is compliant with ROR's policies, the content will be sent for a schema validation check. Once approved, a subsequent preparation for metadata update is sent to the ROR production site for the public. The release of the information is done on a rolling basis and happens at least once every month.

ROR started in 2016, but its main launch years are from 2019 to 2022 with an initiative to specifically focus on affiliation identifications and expand from community use cases to integration into more open scholarly infrastructure [104]. Although registering a ROR ID is completely free and easy to use, its main purpose is focused on affiliation identifications. On the other hand, the project will be based on metadata and data of a more general scope.

Figure 3.4 shows an example of an ROR identifier with full URL representation[13]. The URL of an ROR identifier starts with the ROR domain *https://ror.org* and is followed by a slash that separates the ROR ID from the front [13]. An ROR ID consists of unique 9-character strings and is assigned randomly, so it does not contain organisational information and would not be easily guessed from the content of the strings [13]. An example of the ROR ID would be *02mhbdp94*. The API can recognise three forms of the ROR ID: https://ror.org/02mhbdp94, ror.org/02mhbdp94 and 02mhbdp94 [13].

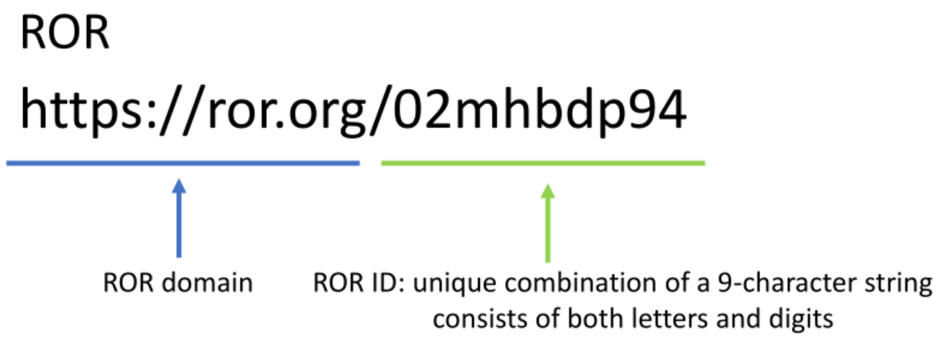

ROR
https://ror.org/02mhbdp94
ROR domain
ROR ID: unique combination of a 9-character string consists of both letters and digits

Figure 3.4: ROR [13]

3.1.4 Archival Resources Key (ARK)

Archival Resource Key (ARK) identifiers aim to provide long-term storage and access to any form of object and are expressed in the form of URLs. They are object identifiers and objects can be in the form of digital, physical, intangible, and living beings and groups [105]. To date, there are 8.2 billion ARKs created by more than 1100 organisations worldwide since 2001 [106]. The use of ARK aims to not return 404 Page Not Found errors and always return the requested items persistently. The creation process can take less than 48 hours to accomplish [107].

ARK is an identifier scheme implemented by the California Digital Library that aims to preserve the object persistently [89]. It started in 2001 and can be referred to as a "specially constructed globally unique, actionable URL" [105, 99]. ARKs are open and decentralised persistent identifiers that require the following items to function: a Name Assigning Authority Number (NAAN), a minter, a resolver, an assignment plan, and an access persistence policy.

ARK is affordable to use at no cost; convenient, self-sufficient, and globally resolvable so that the user can host ARKs on any web server and permits allowance on any resolver [105]. Any user can use their own metadata schema and create an unlimited number of ARKs [108].

When the location of the URL changes, it can cause important content to move around and result in a reduction or loss of information compared to the original copy. The usage of a persistent identifier ensures that interlinking services will not lose information and offers a strong level of stability [105]. A resolver is a system that redirects and forwards the incoming identifier to the current location that hosts the content [105]. ARK relies on a resolver or URL redirection to the current location if the items have changed their location.

Each ARK is unique. Once an ARK-to-object association has been published, that association is considered unique in the indefinite future [109]. ARK can be adapted and enriched with specialised services that provide added functionality [86]. Figure 3.5 shows an example of an ARK URL representation [14].

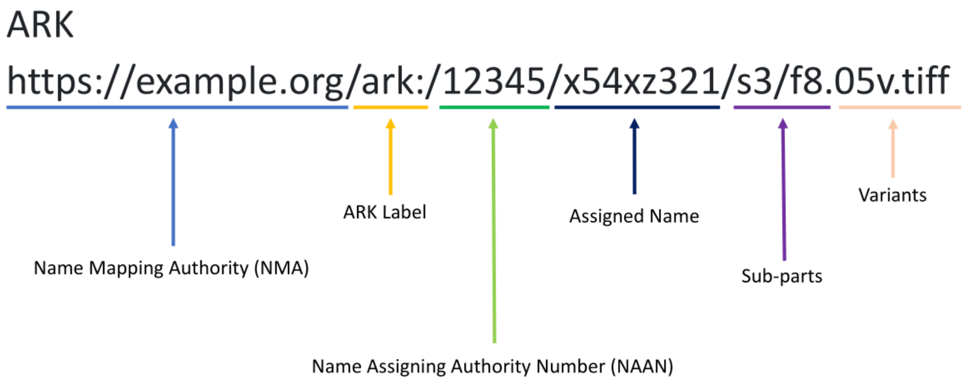

ARK
https://example.org/ark:/12345/x54xz321/s3/f8.05v.tiff
ARK Label
Assigned Name
Variants
Name Mapping Authority (NMA)
Sub-parts
Name Assigning Authority Number (NAAN)

Figure 3.5: ARK [14]

3.1.5 International Standard Name Identifier (ISNI)

International Standard Name Identifier (ISNI) is an ISO standard used by many organisations worldwide. ISNI is a global standard governed by the International Standards Organisation (ISO) and can also be referred to as ISO 27729 [99].

In order to associate with the published work and give credit to the right creators unambiguously despite occurring problems such as name changes, the ISNI International Agency (ISNI-IA) assigns a persistent unique identifying number to eliminate such problems. ISNI spans multiple domains and becomes a crucial key bridge connector between Linked

Data and Semantic Web Applications [110]. The ISNI database is derived from many data sources worldwide and developed using matching algorithms [111].

In 2010, six major organisations worked together, contributed initial data resources and software designs to establish the ISNI International Agency (ISNI-IA) [112]. ISNIs utilise the authority control approach and are primarily managed by organisations such as libraries [86].

ISNI is free to create and controls more than 15 million identities [110]. However, ARK identifiers have more than 8.2 billion identifiers which are also free to use [107].

Figure 3.6 shows an ISNI URL representation [15]. The ISNI consists of a total of 16 digits with 15 digits at the front followed by a check character at the end. The entire URL *https://isni.org/isni/0000000080456315* identifies the entity; while */about* can be added near the end of the URL to identify the description of the entity [15].

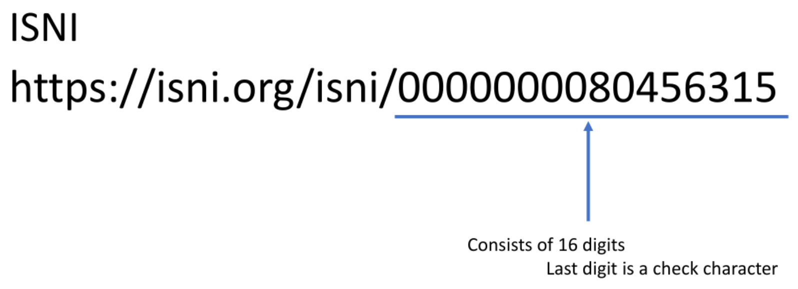

ISNI
https://isni.org/isni/0000000080456315
Consists of 16 digits
Last digit is a check character

Figure 3.6: ISNI [15]

3.1.6 Handle (HNDL) System

The Handle (HNDL) system is a distributed information system designed to provide a global secure naming service for use either as an underlying system or on its own [99]. The Handle system has the infrastructure that allows the association between an identifier and a location [86]. The Handle system has a central registry to resolve URLs to the current location of the content [89]. The Handle system shares the same technical infrastructure as DOIs, so the DOI system also uses The Handle system [89, 113].

The Handle system provides a global name service that can be used securely on the Internet and allows users to use the technology to identify digital content independent of its location [113]. Multiple groups and organisations provide the assignment of the registry, management and maintenance services for the Handle System [86].

Global Handle Registry (GHR) contains records of prefixes assigned to local handle service providers, and GHR's operations are managed by the DONA Foundation and many other organisations called Multi-Primary Administrators (MPAs) [114]. Each MPA is given a credential that is represented by a number. The MPAs have their own authority to assign derived prefixes from their credentials and provide identifier and resolution services for the handlers they allotted.

An example of MPAs is the Corporation for National Research Initiatives (CNRI). The Handle.Net Registry (HNR) is run by CNRI and can assign users a prefix on request; in addition, the user can register with the HNR to enable their identifiers [16]. For every new prefix allotted in CNRI, a one-time $50 registration fee and an annual service fee of $50 are required [115]. The high-cost fee for registration and maintenance is not a good candidate for open-source projects.

The example URL of the Handle System is shown in Figure 3.7 and is made up of a prefix and a suffix [16]. The prefix is assigned to the naming authority and in this case it is the Handle.Net Registry and has the given number 20.500 [16]. Users who have a prefix given by HNR will have the prefix 20.500.xxxx (x) where xxxx (x) is a four- or more-digit number [16]. The suffix refers specifically to the resource, and it is the local name of the resource.

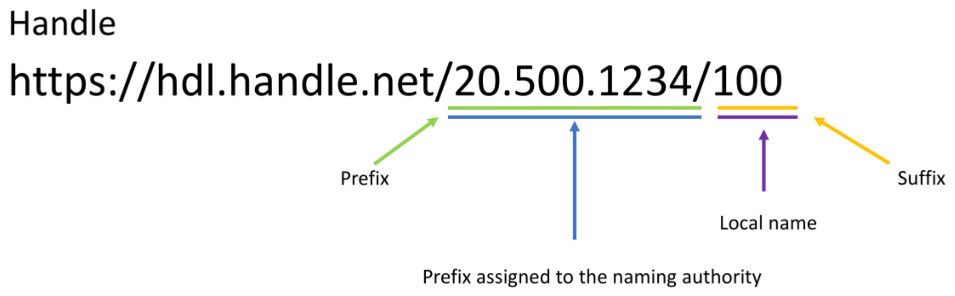

Handle
https://hdl.handle.net/20.500.1234/100
Prefix
Suffix
Local name
Prefix assigned to the naming authority

Figure 3.7: Handle [16]

3.1.7 Persistent Uniform Resource Locator (PURL)

A Persistent Uniform Resource Locator (PURL) is a URL that points to an intermediate resolution service instead of directly to the object and sends the result back as a standard HTTP redirect [116].

A survey conducted by the OCLC Web Survey showed numerical results of the survival rate of normal websites: 13 percent of the IP addresses created in 1998 still survived and lasted in 2002, 19 percent of the sites created in 1999 still lasted in 2002, 33 percent of the sites created in 2000 still last in 2002, and 51

A PURL is a URL that points to the resolution service and not to the actual location on the web of the digital object, a redirection to the requested location of the web resources using standard HTTP status codes [89]. The resolution service maintains and redirects to the current location of the resource, so the PURL is a permanent web address that would redirect to another page [99, 89].

Table 3.1: PURL Type and Meaning [3]

PURL Type	Meaning
301	Moved to a permanent target URL, any request to the old URL will transfer to the new URL
302	A redirection of the old URL to a target URL
303	Use other URLs to access the requested resources
307	A redirection of the old URL to the target URL temporary
404	Gone temporarily
410	Gone permanently

PURL is implemented as an HTTP redirect, so it is only persistent if there is an HTTP server on the host domain name and will not work if the domain changes or ceases to exist [117]. PURL relies heavily on the Domain Name System (DNS) and the Internet [117].

When the website is moved to a new location, the author can update the PURL to point to the new location [3]. Each PURL consists of a target that lists where the PURL redirects and a status code [3]. Some common status codes and their meanings for PURL are shown in Table 3.1 [3].

3.1.8 Uniform Resource Name (URN)

Uniform Resource Names (URNs) are a persistent resource identifier that is not location dependent [118]. It allows for a simple mapping of namespaces into a single URN namespace [89]. All URNs have the syntax shown below:
“URN” “:” NID “:” NSS
where NID is the Namespace Identifier, and NSS is the Namespace Specific String [17]. NIDs are case-insensitive and the NSS is a unique string within the URN namespace; the result of the combination of NIDs and NSS makes the resulting URN globally unique [17]. Figure 3.8 shows an example URN with the “URN” “:” NID “:” NSS format mentioned above [17].

The object to which the URN points could have stopped or might not be available, but the URN will remain globally unique and persistent [89].

3.1.9 Organization Identifiers

Organisation identifiers aim to create and enable a connection link between organisations such as research institutions, funders, corporations, etc., and the objects [99].

Organisations have very different hierarchical internal structures, different sizes, and dimensions, and could undergo reconstruction as time goes on [86]. Funding is an important part of research institutions. ORCID, DataCite, and Crossref enable funding awards to be linked to the contributor, the date of the contribution, and the awarded publication entities.

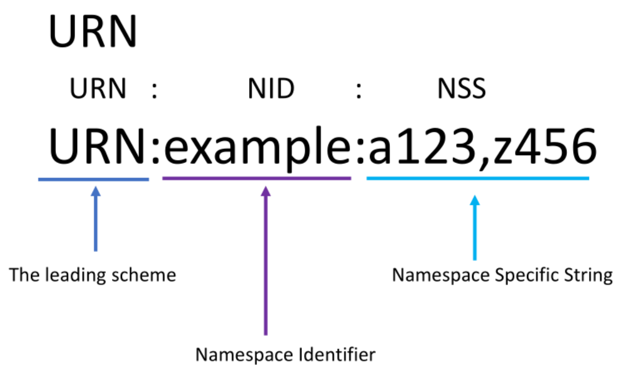


Figure 3.8: URN [17]

This allows future users to have a reference of the application process, as well as to improve and track the impact of research investments [86].

Two of the examples of organisation identifiers that will be explained in detail are Funder ID and GRID ID. Funder ID, listed in the Crossref Funder Registry, is an open registry of persistent identifiers that links grant and funder information towards a more transparent funding process [99]. The Global Research Identifier Database (GRID) collects the addresses of institutions and assigns a PID to the corresponding research institutions [99].

Both Funder ID and GRID ID focus on a very specific area of PID. These types of PID system are not well suited for creating PIDs to point to data collections and datasets, in part or in whole.

3.2 Underline Communication System

All of these PIDs rely on a common communication system based on Internet protocols. In this section, a broad overview of computer networking, the Internet, and REST will be provided.

3.2.1 The Internet

The Internet is a widely used medium for publications due to its convenient and fast access to research materials and low-cost distributions [119]. Unlike physical copies of the research publication, electronic copies can be found almost instantly when the correct hyperlink is typed. Furthermore, physical copies often are distributed over printed versions of the publication or paper copies, which makes the transfer of knowledge to be at least the price of the printed content. Electronic copies of the content are distributed on the Internet via hyperlinks.

Hyperlinks are a good method for locating the desired information because they can identify the research information unambiguously and preserve the publications stably over a long period of time, even when there is the possibility of transfer of authority and change of formats of the content [119]. Billions of users operate on different client-server implementations that access the Internet using their hardware devices on a daily basis [5]. An example of requesting a *picture.gif* from the website is shown in Figure 3.9 [18], where *www.someSchool.edu* is the hostname, */someDepartment/picture.gif* is the path name */someDepartment* is the directory name, and *picture.gif* is the file name.

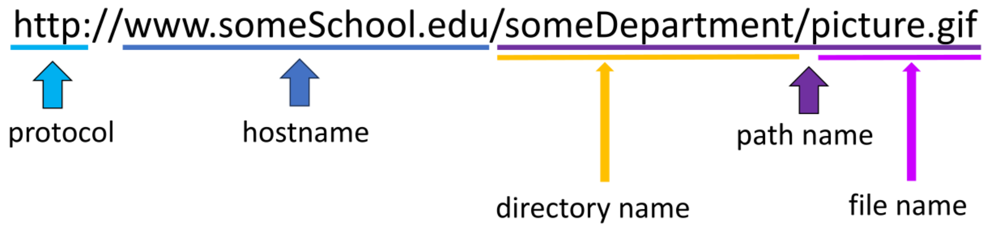


Figure 3.9: An Example of URL [18]

The Internet is a computer network that connects numerous devices around the world. In this section, the Internet is introduced by summarising the book by Kurose and Ross [4]. For other in-depth understanding, please refer to their book for more details.

The devices that are connected to the Internet are also called hosts or end systems in the Internet jargon. It is important that all devices follow certain standards and rules to allow the success of the flow of information exchange. Protocols are applied to all end systems to control the sending and receiving of information via the Internet. The protocols and Internet standards need to be obeyed, agreed, and followed by all the devices in order for further systems using the Internet to work interoperably throughout the world.

The Internet standards are developed by the Internet Engineering Task Force (IETF). Request for Comments (RFC) is the name given for the IETF standards documents. RFCs contain the protocol for the end systems. The protocol defines the format and order of messages exchanged between two or more communication devices, as well as their actions in transmission, during the receiving stage, and during other events. Applications are assigned specific port numbers on the Internet. The default port number for HTTP is port 80 which is unencrypted and port 443 for HTTPS connections, which is more secure.

3.2.2 Hypertext Transfer Protocol (HTTP)

Uniform Resource Locator (URL) is a form of Uniform Resource Identifier (URI). URI was introduced and published by Tim Berners-lee in 1994 [119]. URL uses the HTTP protocol to locate resources hosted on web servers [119].

Uniform Resource Locator (URL) protocol includes the Hypertext Transfer Protocol (HTTP) [88]. Searching for a particular object is much easier with the help of search engines and Hyperlink [4]. URL can be used to create hyperlinks to view documents on the World Wide Web (WWW) [119]. The URL standard was first defined in RFC 1738 [5].

The process of how an Request for Comments (RFC) starts from a draft to its finalised version is summarised in the book by Richardson, Amundsen, and Ruby [5]. An RFC started off with a submitted Internet-Draft through the process called the Standards Track. The Internet-Draft has a lifespan of 6 months. After six months, there are three ways that can happen to the Internet-Draft. It can either be approved and become an official RFC, change its content and resubmit, or expire. If the Internet-Draft is approved, the draft expires immediately and is replaced by the official RFC. If changes are required, an updated version can be re-submitted. If neither of those two actions takes place, then the Internet-Draft expires and can no longer be used.

In the following section, HTTP is introduced by summarising the book by Kurose and Ross [4]. For other in-depth understanding, please refer to their book for more details.

HTTP defines the format, structure, and sequence of messages sent between the browser and the server, as well as how messages are communicated between the server and the client. If an Internet user follows the HTTP protocol, then they can retrieve a website from any web server that also follows the HTTP protocol.

The requested web objects are hosted on the server and can be retrieved by the user via a URL. HTTP defines how the web user can request their desired web objects from the web servers and how the server can transfer the objects back to the user. It could start with the action of clicking on a hyperlink as a form of sending a request to the web server. The browser sends an HTTP request for the objects it wants. The server receives the request and sends back a response message. If the response message says successful, then the requested objects will also be sent back to the user. Figure 3.10 shows the process of the clients sending an HTTP request to the server and the server sends back the HTTP response [19].

HTTP/1.0 is the original version of HTTP that is being used. Web server users started using HTTP/1.0 in the early 1990s. It was later standardised in 1997 and the protocols regarding HTTP/1.0 can be found in RFC 1945. Currently, there are newer versions of HTTP being used. Although a majority of the HTTP transactions in 2020 are still using HTTP/1.0 and follow RFC 7230, HTTP/2.0 has gained its popularity and more and more web server users are starting to use the newer version. HTTP/2.0 is included in RFC 7540 and is standardized in 2015. Over 40 percent of the top 10 million websites support HTTP/2.0. HTTP/2.0 does not change the general format of the request message and the response message, but changes how the data are formatted so that the requested message or object can transmit faster.

There are two types of HTTP formats: request message and response message.

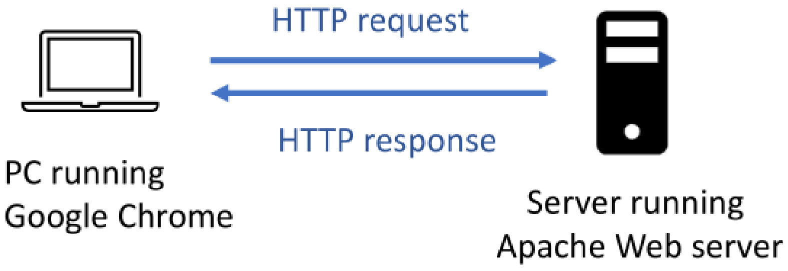
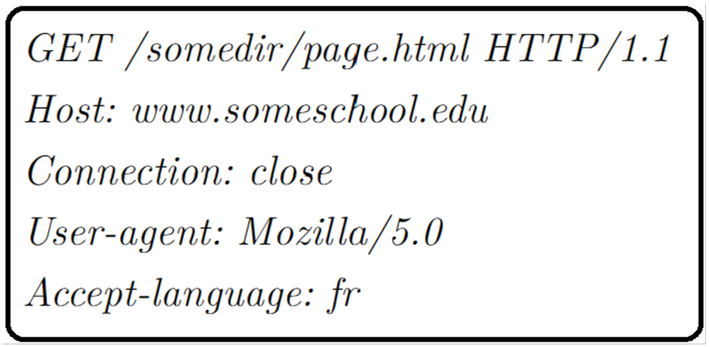


Figure 3.10: HTTP request-response behavior [19]

Figure 3.11 shows an example of an HTTP Request Message, it is taken from the book by Kurose and Ross [20].



```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Figure 3.11: HTTP Request Message [20]

The message is transmitted in ASCII text, so it is both readable for machines and for human beings. The response message can be as small as one line or many more lines for a more detailed response. The first line of the response message is the request line, followed by the header lines.

The request line usually consists of three fields: the method field, the URL field, and the HTTP version field. Some of the commonly used method fields are: GET, POST, HEAD, PUT, and DELETE. In the provided example, the request line used the GET method to request an item called page.html from the directory /somedir using HTTP/1.1.

The header lines start from the second line of the response. The host shows that the requested object is hosted on the website www.someschool.edu. The Connection states close, which means that it wants the server to close the connection once the object is sent, so there will be no persistent connections. The User-Agent shows that the Internet browser used is a Mozilla Firefox browser version 5.0. The Accept-Language shows that the user wants the

Table 3.2: HTTP Methods

[4, 5]

HTTP Methods	Description
`GET`	Get a representation of the resource
`HEAD`	Similar to the GET method. Receive the header lines and not the content of the requested object. HEAD is used when the user wants to save bandwidth or test to valid information retrieval and modification
`PUT`	Modify a part of the resource with a new given representation and send back the representation back to the server
`DELETE`	Delete the representation of the resource
`POST`	Create a new representation of a resource given its content
`OPTIONS`	Give back a list of the HTTP methods that the resource can respond to
`PATCH`	Modify part of the state of the resource, similar to PUT but allows application of a more fine-grained change
`LINK`	Connect some other resources to the current resource
`UNLINK`	Destroy the connection of some resources from the current resource

response back in French; if the language is not specified, then the server will send back the requested item in the default language of the browser. There is also content from the Request message called entity body. The entity body is empty in the provided example because the GET method is used. If the POST method is used, the entity body will be filled out with the content that the user posts.

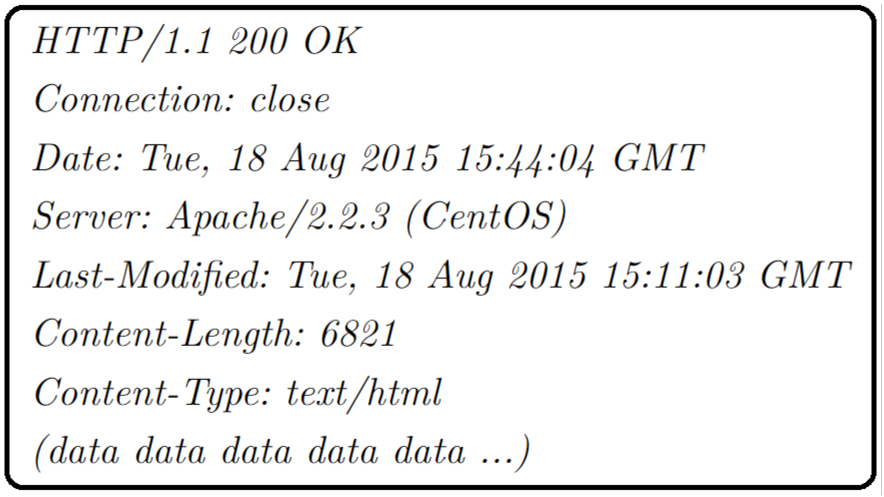
POST method is used when there is a search engine or if there is a form to be filled out. Similarly to the method GET, the method POST still sends requests to retrieve the web page, but the content of the website depends on the information filled out in the form. The entity-body of the HTTP Request Message will show the content that the user enters in the form.

In table 3.2, the HTTP methods are introduced and discussed. The table is summarised from the book by Kurose and Ross and from the book by Richardson, Amundsen, and Ruby [4, 5].

Idempotence is mentioned in the book written by Richardson, Amundsen, and Ruby [5]. When the Internet experiences a time out and the response is not sent back from the server, the user does not know the exact situation and could send the same request again. The command could have been executed twice by the server. Some HTTP methods are idempotent, which means that the state of the resource does not change after the first time the command has been sent. DELETE, as an example, is idempotent. When the same

DELETE command is sent to the server several times, the resource state will be exactly the same as it is after the first request is executed. The PUT method is also idempotent, so even if the same PUT command is sent multiple times, no extra changes will be applied after the first change took place. The GET method is also idempotent. The POST method, on the other hand, is not idempotent. If the same command is requested a few times, the same content is going to be created multiple times, each with a different created time stamp.

Figure 3.12 shows a typical HTTP Response Message and it looks like the following; the example response is taken from the book by Kurose and Ross [21]. The response message consists of several sections: a status line, several header lines, and the entity body.



```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

Figure 3.12: HTTP Response Message [21]

The status line consists of three fields: the protocol version field, a status code, and a corresponding status message. The response message shows that it is requested over HTTP/1.1 and the requested content is found and will be sent back. The entity-body at the end of the message is the content that is sent back to the user, represented by (data data data data data ...).

The header lines will show more details about the message, but not the message itself. The Connection indicates that it will close once the requested item is sent back, showing that no persistent connection is needed. The date shows the exact date and time that the item is sent by the server. The server shows the response is generated and sent over an Apache Web server. The Last-Modified shows the last time the object was modified. The Content-Length and the Content-Type indicate the number of bytes of the requested object and the format in which it was sent back.

The following sections are summarised from the book by Richardson, Amundsen, and Ruby [5]. The response headers are sent after the status code, followed by the entity-body, which contains the main content of the response. One of the response headers is the Content-Type, which can also be viewed as the media value of the entity-body. Some of the commonly

Table 3.3: Response Code [4, 5]

Response Code	Description
200	OK; Request is sent successfully and the response will be sent back soon
301	Moved Permanently; The requested object has been moved permanently
400	Bad Request; The request can not be understood by the server
404	Not Found; The requested object does not exist on the server
505	HTTP Version Not Supported; The requested HTTP version is not supported by the server

used Content-Types are text/html for HTML, image/jpeg for images. One of the special types of the Content-Type is application/vnd.collection+json.

JSON is a standard for representing a type of data structure in plain text. It is defined and represented in RFC 4627. Plain text needs to follow a set of data structures in order to be classified as JSON: double quotes for strings, square brackets for lists, and curly brackets for objects. Application/vnd.collection+json strictly follows the rules, and it is a collection of object(s) that contain item(s) that contain list(s) that contain object(s) within. An example of the Application/vnd.collection+json would look like the following.
{“collection”: {“items”: [{}, {}, {}]}}

The status code can also be called the response code. It is represented as a three-digit number that summarises what the response is from the server after a request is sent.

Table 3.3 discusses the HTTP response code, it is summarised from the book by Kurose and Ross and from the book by Richardson, Amundsen, and Ruby [4, 5].

RESTful Web Services

REST stands for Representational State Transfer and is a formal design for web servers that are built with the purpose of supporting the functionality of sites or applications [120]. API exposes and interacts with large amounts of data between computer programmes to exchange information, so client programs use the API to communicate with web services [120].

Unlike a majority of the APIs these days that remain unchanged and static, the representational state transfer (REST) API is designed to manage changes. REST is a set of design constraints and was first introduced by Roy T. Fielding in his PhD dissertation for software architecture [5]. In Web APIs, messages are sent to and from following the HTTP protocols; while for the RESTful system, messages also need to follow a predefined protocol [5].

REST is a set of guidelines on how resources are designed over HTTP [121]. An HTTP message is composed of four main components: verb/method, resource path, headers, and body [121]. REST has become a widely used standardised way of publishing services using its specialised designed architecture aimed at supporting interoperability between different computer systems on the World Wide Web (WWW) [122]. To achieve improved performance and make the text easily readable by machines across the network, RESTful APIs use HTTP Methods such as GET, POST, DELETE, and PUT as part of the design constraints [122]. The requested object can also be called a resource. A resource is identified by a unique Universal Resource Identifier (URI) [123]. The interface supports the HTTP common operations and allows the resources to have different representations, e.g. text, xml, json, etc [124]. Out of the mentioned formats for resource representation, JSON is one of the most popular and widely used formats. REST APIs can handle many data formats, and the JSON format is set to be the default format for REST APIs [125]. JavaScript Object Notation (JSON) is an open standard file format that aims to provide human-readable data interchange [124].

Statelessness is one important characteristic of REST. The application states are the states regarding the client request which may contain a history of interactions with the server and their current context information [124]. The application states are stored on the server side. Statelessness refers to the server reacting to the client's request regardless of the application states.

Web service is a software service that communicates between computing devices using standardised communication protocols through the World Wide Web (WWW) [124]. Software can be used by other software applications through an interface called the Application Programming Interface (API) [126]. API provides an interface that different applications can easily interact and integrate with other software systems [126].

REST is an architectural design that provides guidelines for distributed systems to communicate using existing Web protocols to create web services and APIs [124]. The RESTful API uses the HTTP protocol. The HTTP protocol is stateless and does not store any previous information about the client, so each request sent from the client must include all the information needed for processing [121]. Figure 3.13 shows an example of a REST API [22].

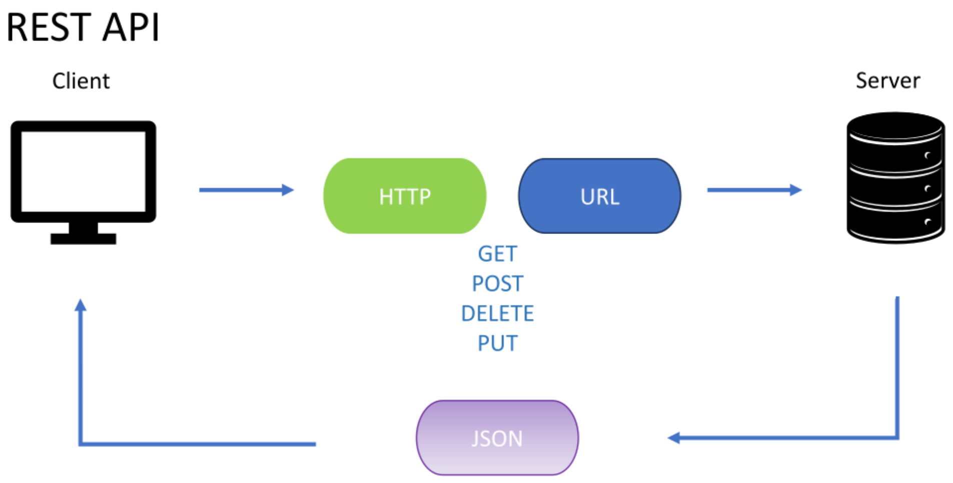


Figure 3.13: REST API [22]

Chapter 4

Prototype Design

Persistent identifiers provide stability of the information stored despite the movement of content between websites [127]. Without the usage of persistent identifiers, users need to be physically informed about the new link to keep using the needed content; thus avoiding broken links. Persistence provides users with the stability to store and locate data so that individual use and organisations can focus more on the quality of finding desirable data and processing the existing data. Persistence provides users with continuous access to the data. Data usage in research requires particular long term storage for data processing as well as if other researchers decide to use the same data to conducting same or further experiments on a much later date. The most used persistent identifiers were discussed in the previous chapter. Of all the mentioned persistent identifiers, DOI, ORCID, and ARK are the top candidates for our PID system because they are very widely used and can be very easily integrated and accessed through databases.

After thorough consideration, ARK was chosen as the persistent identifier implementation for our prototype. DOI is a persistent identifier paid for that is used for scholarly communication. It is one of the most popular and widely used persistent identifiers existent with over 300 million DOIs existing. Registering for a new DOI name differs in price because each different registration Agency offers different services. ORCID is financially sustained by membership fees from organisational members; however, participants in research, scholarship, or innovation can register an ORCID ID for free. Unlike DOI and ORCID, which require fees for issuing new persistent identifiers, ARK is completely free and open source. The relatively short creation time for ARK persistent identifier of less than 48 hours is also one of the reasons we choose to use it.

ARK is a PID protocol that we can extend to allow the user to download specific data within the dataset. The design is constructed in the form of a PID URL to download specific parts of a dataset. For the user, no new software for writing code is needed - only a browser and clicking on the PID link. An interactive design that extends the prototype is used to allow for more specific data queries. The design would start with downloading the file in the dataset and then improving to download only specific columns in that file.

4.1 Using ARKs as Persistent Identifiers (PIDs)

The following sections discuss the structure of ARK and how ARK operates as a persistent identifier (PID).The majority of the sections are summarised from the ARK Identifier Scheme by Kunze [23].

Founded in 2001, ARK is an open and decentralised persistent identifier that uses web address in the form of URLs to identify anything digital, physical, or abstract [107]. The benefit of using ARK as persistent identifier is that it will link directly to the item the user is requesting without stopping on a landing page and it will not return a 404 Not Found page [107]. There are specific requirements to prevent this from happening.

There are three requirements for an ARK mentioned in the ARK Identifier Scheme [23]. The first requirement of an ARK is that the link will point to the object and there will be supervision and control over the object. The second requirement is that the link provides a description of the object. The third requirement is that clicking the link will lead to pointing to the object or copy of the object.

An ARK in the form of URL is shown in Figure 4.1 ARK Anatomy [23]. The ARK Anatomy is summarised from The ARK Identifier Scheme which is composed of many different parts and can be extrapolated and explained in further detail [23]. The first way to represent an ARK is to attach a resolver service with a Compact ARK. The Resolver Service varies in different formats and in different organisations, but should be compatible with the Compact ARK, which is globally unique. The ARK can also be divided as prefixes, base name, and optional suffixes. Prefixes consist of the Name Mapping Authority (NMA), Label, and the Name Assigning Authority Number (NAAN). The base name consists of the shoulder and the blade. Lastly, the suffixes consist of parts and variants.

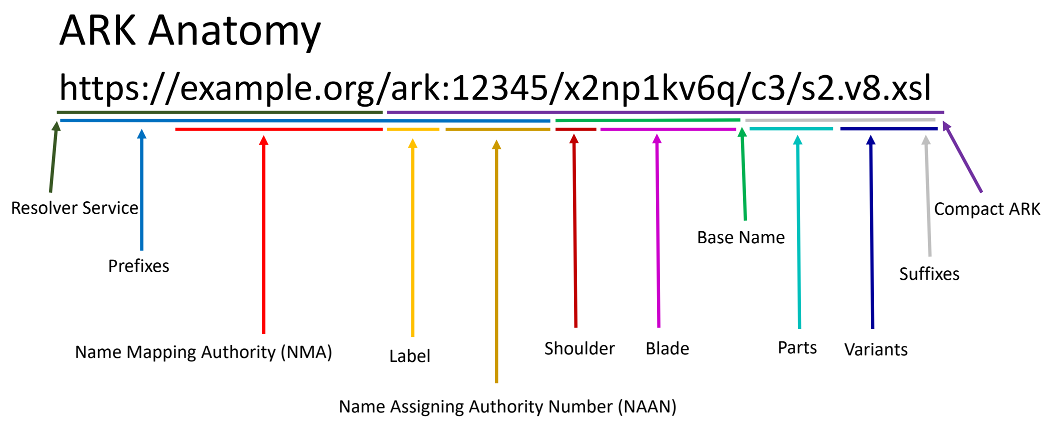

ARK Anatomy
https://example.org/ark:12345/x2np1kv6q/c3/s2.v8.xsl
Resolver Service
Prefixes
Name Mapping Authority (NMA)
Label
Name Assigning Authority Number (NAAN)
Shoulder
Blade
Base Name
Parts
Variants
Suffixes
Compact ARK

Figure 4.1: ARK Anatomy [23]

The Name Mapping Authority (NMA) is an Internet hostname and has the same format as the hostname of a URL that can be referenced in Figure 3.9. The important part of NMA is that it is "identity inert" if it is followed by the same Compact ARK even if the NMA differs [23]. They are all just different derivations of the same original object represented by the Compact ARK. The resolver is a website that forwards the incoming identifiers to the website that currently holds the information in the process called resolution and can also be called redirection [127]. The hostname, as shown in the URL representation in Figure 3.9, should be deliberately chosen to be persistent enough for the URL to last a long time. The Label shows and signifies that the URL is an ARK identifier. When a request associated with the NMA's identifier is sent in, the NMA usually takes action and gives out some responses: whether the response is to provide access to the object or forward the request to some other NMAs [128].

The Name Assigning Authority Number (NAAN) indicates and specifies the organisation such as an institution, a department, a laboratory, etc. that originally assigned the name for the object [23]. The combination of the Label ARK and the NAAN is immutable [128].

The resolver redirects the ARK to the URL where the resource currently resides [129]. A root resolver such as N2T.net can work well with many different types of identifier [128]. N2T.net stands for Name-to-Thing, which is a meta-resolver that redirects requests to the server that contains the content [130]. N2T.net works compatibly with more than 900 different identifier types, treats all identifiers equally, and provides the same high quality services to all of them regardless of their types [130]. There exists a special feature of the software technique called content negotiation, which can request another format form that is different from the original form of the object, e.g. requesting a PDF form of an HTML file [131]. In addition to that, N2T supports "cross-scheme features" that allow different identifiers to work together, such as "[returning] ARK-style inflections for DOIs, and ARKs [to] return DataCite DOI metadata via content negotiation" [130]. In addition to being a metaresolver, N2T.net also stores more than 50 million identifiers on its site [130]. When an incoming ARK string is sent to N2T.net and N2T.net cannot locate the ARK string in its database, N2T.net will redirect it to the local resolver associated with the NAAN [132]. Inversely, the local resolver can also be set up for configuration to forward unknown ARK string back to N2T.net [132].

The base name on the ARK Anatomy consists of shoulder and blade. The shoulder is an extension of the NAAN and identifies the subnamespaces [128]. A new sub-namespace is created when a prefix associated with an ARK namespace is extended [132]. There could be one NAAN with multiple shoulders. Each combination of the NAAN with that particular shoulder is unique. Although NAAN and shoulder remain fixed for one particular subnamespace, the blade is different for all identifiers [128]. The shoulder is an optional part of the ARK anatomy. When there are multiple internal departments or units under the NAAN, shoulder usage is preferred and reserved for the individual unit [23]. Organisations have many different units or projects underneath them. Shoulders are non-overlapping namespaces with precise and determined strings extended to the broader organisation which is represented as the NAAN in ARK [132]. Setting up a new shoulder for every new project or newly issued unit will benefit the NAAN as the assigned ARK name will never conflict or overlap with existing ones [132].

The base name of the ARK needs to be uniquely identified so that no duplicates of the same name are assigned. A tool like a minter can generate unique strings for the resource object, manage the metadata about the object while keeping track of the existing assigned strings to ensure there are no duplicates [129]. Nice Opaque Identifier (NOID) functions well as both a minter and as a name resolver. NOID has a database that will check and ensure that no identifier is minted twice. During the process of creating the identifiers, NOID can choose to add a specific selected character at the end of the string to later check for transcription errors [133]. The NOID-generated ARK string has the feature called the NOID Check Digit algorithm (NCDA) that checks the last digit of the base name to ensure that the combination of NAAN and the base name is accurate and correct [128]. NOID cannot generate ARK directly but can generate unique strings that can be used for the base name [133]. When joined with the Resolver Service and the Label, a working ARK is created. The usage of a name resolver allows the object to be associated with a published persistent URL that never changes, and the actual location of where the object resides can change. When NOID functions as a local name resolver: it maintains a lookup table for redirection of the persistent URL and redirects the link to the object [133].

The Compact ARK is made up of strings of ASCII characters which can be composed of letters, digits, and the following seven special characters [23]:
= ~ * + @ _ $

The following four characters can also be used but they have reserved meanings and should only be used when intended [23]:
% - . /

If the two characters / or . appear as the last character in the ARK, then it has no meaning and should be ignored; if they are in the middle of the ARK, then it can be used to distinguish the hierarchy of objects and variants of objects [23]. If the two special symbols are used in ARK, they have special meanings: "/" indicates containment, and "." indicates variation [128]. The character % is reserved for encoding octets in the ARK string [23]. Many text formatting processes would introduce hyphens in the process and treat hyphens as significant characters, which would lead to misrepresenting the original persistent identifier string [131]. In the ARK representation, the hyphens are identity inert, which means that the ARK string with the hyphens is the same as the same ARK string without the hyphens [131].

Lowercase and uppercase letters are two different representations in ARK and are treated separately, so there is the possibility of creating a shorter ARK representation with the combination of lower and upper case letters; however, to avoid confusion, it is best to use only lower case letters [131] .

Some considerations might have to be taken care of before or during an ARK is created such as: an access persistence policy needs to be taken into consideration to choose what methods can securely preserve the data, what kind of variant qualifiers should be used so they can offer the desirable services to the resources, what kind of component qualifiers should be used to reveal whether just part of the relationship or the whole relationship, the metadata standards to apply to the data, and whether a suffix passthrough should be used if many subresources are containing a large number of files under the resources [129].

4.2 PID Design

Previously we defined and elaborated on the ARK specification for our PID design, further we introduced the concept of RESTful Web services, which forms the underlined protocol used when using PIDs. RESTful web services are used to communicate between the user and the n2t.net server.

The intention is to create a PID system that points to specific data from a given data repository. The prototype should return data using URLs that the user requests. The user can also send the request type to tell the server what to fetch. The RESTful Web Service command GET is used. One of the reasons why only GET is used is because GET is idempotent, meaning that if the server will output the same result if GET is used more than once in the database. If the server is having connection problems, the same command GET is executed more than once, the output will not be affected. The second reason why GET is used is that the data sets should not be modified by anyone trying to access the data. The data should be protected and are read-only, so the data keep being consistent and are reproducible for open science.

The majority of the work in this section was first presented at the 16th International Conference on Metadata and Semantics Research (MTSR) at the University College London in London, UK, between 7 and 11 November 2022 [37].

As described in Chapter 2, the Federated Catalog Registry provides the ability to search the description of the data and related resources. A Federated Metadata registry is an extensive database that stores and manages datasets in a given repository. Within the registry, a Federated Data Catalog system provides a single point of access across all data regardless of the location the data are stored. The use of ARKs as persistent identifiers means that specific data can be queried and accessed from a request in the form of a URL.

As mentioned in Chapter 2.1.1 Catalog Metadata Registry (CMR), the Data Catalog behaves like a data storage for metadata and serves with some data management tools for data analysis. A metadata Catalog System provides a single place where all data can be catalogued, enriched, searched, tracked, and prioritised. Data catalog is a collection of metadata, combined with data management and search tools; the combination serves as an inventory of data and provides tools for data analysis. The user can search and find data quickly, when all available datasets are made available in one platform. Catalog Metadata Registry (CMR) enhances data management and accelerates data-driven decision making by providing an effective unified platform to host all the data required for making a prone decision.

Core is the main classifications and groups among the Catalog Metadata Registry. Core is the metadata that are common in all datasets; e.g., dataset name, size, type of data. The function of exporting the core metadata given the dataset is incorporated in our design and will be elaborated in a later section. Domain is the metadata for a particular field or area. It can be a category or a group of data elements that share the same purpose or meaning. The future prototype wishes to have the ability to connect different categories together to establish a connection between each other. The relation is information about another resource that is related to the current resource. The relation can be entered into the metadata form in two formats: dropdown menu or text entry. Using timestamp as the primary key can link different datasets with the same timestamp to see the correlation between energy consumption and weather information.

As mentioned in Section 2.2.1 Types Metadata Registry (TMR), a metadata registry is a database used to store, organise, manage and share metadata [58]. Issues encountered when building the extensive metadata registry are that participants may have used diverse schemas and description methods to create their metadata records [85]. Building one integrated metadata registry simplifies and enables one search for the end-user to retrieve the information needed rather than searching multiple times through all the different existing individual databases. Therefore, interoperability will be a critical component in such development.

Persistent identifiers can efficiently and reliably retrieve the cited work when most data is stored in highly decentralised locations across many online databases. URL is a common format and a way to create a link to the resource. PIDs are implemented in the research workflows to enable a trusted digital net between objects, contributors, and organisations, which alternatively increases the interoperability between resources. A persistent identifier is a unique identifier pointing to an entity unambiguously and often represented as a string of letters and numbers.

The prototype was designed to respond to requests in the form of URLs. By applying ARK structure to the URL format allowing users to have a persistent identifier that is internet-based. The ARK PID for the prototype follows a specific pattern. First, we need to understand how the datasets are structured. The datasets contain different files, and the PID needs to also know what kind of types are contained in the file. Once an overall understanding of the types contained is achieved, extraction of the data becomes much easier, in whole or in part. An example of the URL would be:

http://ARK.org/<store>/<partition>/<sector>

where store can be a dataset, partition can be a file.

The use of PID is seen as a part of the solution to the problem of reproducible research and science. This section concludes a preliminary method using PID to reproduce research results using time-series data. The prototype uses time-series data for design, implementation and testing, with continuation on the methodology and design, other types of datasets can also be included. Automated processing of large volumes of data requires implicit details about the data types. Data federation combines all the autonomous data stored to form one extensive database. The user could access different types of database servers and files with various formats from all data sources and be accessible through multiple APIs. With the help of the N2T.net resolver service in combination with ARK, the data used for our experiments are stored under the ARK name assigning authority number for The SFU Computational Sustainability Lab, which is 57460 [134].

The prototype follows the HTTP format. HTTP defines and regulates the format, structure, and sequence of messages sent between the browser and the server, as well as the way messages are communicated between the server and the client. The prototype strictly follows the HTTP protocol. Any user can retrieve the same data from a web server that also follows the HTTP protocol. REST is a set of guidelines on how resources are designed over HTTP. It is important that the design follows certain standards and rules to allow the success of the flow of information exchange. The protocol defines the format and order of messages exchanged between two or more communication devices, as well as their actions in transmission, during the receiving stage, and during other events. The default port number for HTTP is port 80 which is unencrypted.

Representational State Transfer (REST) is mentioned in Section 3.2.2 Hypertext Transfer Protocol (HTTP). API exposes and interacts with large amounts of data between computer programmes to exchange information, so that the client programs use API to communicate with web services. REST is a set of design constraints. REST is a set of guidelines on how resources are designed over HTTP. REST has become a widely used standardised way of publishing services using its specialised designed architecture aimed at supporting interoperability between different computer systems on the World Wide Web (WWW). REST APIs can handle many data formats, and the JavaScript Object Notation (JSON) format is set to be the default format for REST APIs. REST is an architectural design that provides guidelines for distributed systems to communicate using existing Web protocols to create web services and APIs. The RESTful API uses the HTTP protocol. The HTTP protocol is stateless and does not store any previous information about the client, so each request sent from the client needs to include all the information needed for processing.

There are multiple ways to represent an ARK. The first way to represent an ARK is to attach a resolver service with a Compact ARK. The resolver service can vary in different formats and in different organisations, but the Compact ARK is globally unique.

Time-series data is a collection of observations made chronologically. Sensor measurements are recorded over a period of time, usually continuously at the time of measurement. The length of time and the frequency of the reading can result in a large amount of data and can have a high dimensionality if the reading records more than one value, observation, or measurement [38]. Each value of each sensor can be marked with a timestamp - the date and time of reading — which is a unique identifier. Since the timestamp is unique, we can use it to reference a specific reading. The row number cannot be used as a unique identifier because it can change based on data manipulation and information processing functions such as row sorting.

To better illustrate our PID design, we will provide examples that pertain to the data in the Almanac of Minutely Power dataset (AMPds) [135]. The dataset contains electricity, water, and natural gas metering data. Sensors measure different consumption characteristics. In each CSV file (one per sensor), the timestamp (per minute) is the unique identifier or primary key.

An example of a PID can be shown as:

Ex.1: https://n2t.net/ark:/57460/AMPds/DWE/V@13332∼13400

where https is a secure HTTP RESTful request, n2t.net is the global resolver, ark means the URL is an ARK, 57460 is the NAAN for our lab, AMPds is the name of the dataset, DWE is the meter or sensor, V is the measurement (column), and @13332∼13400 is the timestamp range of the readings (rows). To select all the readings for a given sensor (e.g., DWE or dishwasher meter) for a given measurement (V or voltage), the following URL would be used:

Ex.2: https://n2t.net/ark:/57460/AMPds/DWE/V@*

To return multiple values (e.g., V for voltage and I for current) we use:

Ex.3: https://n2t.net/ark:/57460/AMPds/DWE/V+I@*

To return multiple values at multiple timestamp (i.e., ID) ranges we would use @13332∼13400+24300∼25500:

Ex.4: https://n2t.net/ark:/57460/AMPds/DWE/V+I@13332∼13400+24300∼25500

Lastly, using cross-fold validation is important for training and testing learning algorithms. To provide the features for testing cross-validation, data can be divided into the requested chunk:

Ex.5: https://n2t.net/ark:/57460/AMPds/DWE/V@*/3

Here is an example when we want to exclude a range of timestamps:

Ex.6: https://n2t.net/ark:/57460/AMPds/DWE/V@@24300~25500

In this case, we are requesting all data except for the data in the inclusive timestamp range of @@24300~25500. The double-at @@ is used as the *exclusion* rather than the minus sign, as this avoids potential URL corruption with text formatting software where hyphens are used for line breaks.

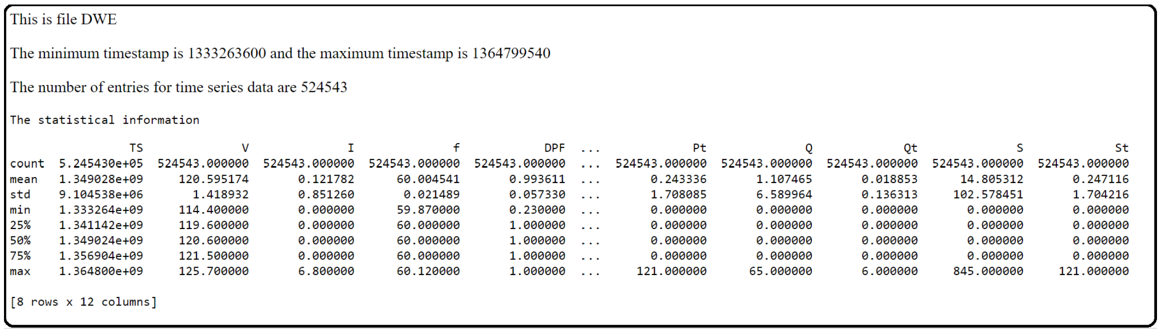
4.3 Prototype Design

To demonstrate our PID design, a prototype was written in Python 3. Python has a wide range of library collections that offer extensive functionalities in the area of data analysis, machine learning, and task automation. We used test data from the AMPds dataset. The AMPds dataset contained measurements of electricity, water, and natural gas at one-minute intervals [136]. The following files from the AMPds dataset were used for testing: B1E.csv, B2E.csv, BME.csv, CDE.csv, CWE.csv, DNE.csv, DWE.csv, EBE.csv, EQE.csv, FGE.csv, FRE.csv, GRE.csv, HPE.csv, HTE.csv, OFE.csv, OUE.csv, TVE.csv, UTE.csv, WHE.csv, WOE.csv. Those files had the same columns: TS, V, I, f, DPF, APF, P, Pt, Q, Qt, S, St.

The AMPds dataset was stored locally on my computer, which also ran a localhost web server. A TOML configuration file was used to store the directory structure of the datasets. TOML stands for Tom's Obvious Minimal Language. TOML served as a minimal configuration file and could be easily and conveniently implemented using the Python language [137].

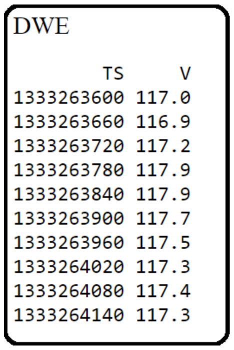
As part of the TOML configuration file, selective files from AMPds with CSVextension were included. The primary keys were also listed in the TOML file to avoid confusion because the primary key was unique to each dataset when multiple datasets were used.

Once the dataset was selected, the CSV files could then be selected. Only one CSV file could be selected at a time. To continue with the file selection, DWE.csv was chosen for demonstration. An input of the URL: http://localhost:8000/AMPds/DWE would give the following result, as shown in Figure 4.2.



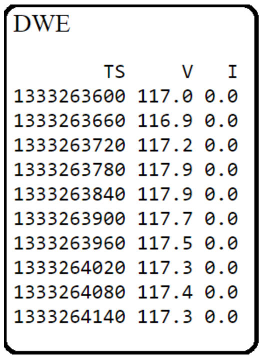
```
This is file DWE

The minimum timestamp is 1333263600 and the maximum timestamp is 1364799540

The number of entries for time series data are 524543

The statistical information

                 TS              V              I              f            DPF  ...             Pt              Q             Qt              S             St
count  5.245430e+05  524543.000000  524543.000000  524543.000000  524543.000000  ...  524543.000000  524543.000000  524543.000000  524543.000000  524543.000000
mean   1.349028e+09     120.595174       0.121782      60.004541       0.993611  ...       0.243336       1.107465       0.018853      14.805312       0.247116
std    9.104538e+06       1.418932       0.851260       0.021489       0.057330  ...       1.708085       6.589964       0.136313     102.578451       1.704216
min    1.333264e+09     114.400000       0.000000      59.870000       0.230000  ...       0.000000       0.000000       0.000000       0.000000       0.000000
25%    1.341142e+09     119.600000       0.000000      60.000000       1.000000  ...       0.000000       0.000000       0.000000       0.000000       0.000000
50%    1.349024e+09     120.600000       0.000000      60.000000       1.000000  ...       0.000000       0.000000       0.000000       0.000000       0.000000
75%    1.356904e+09     121.500000       0.000000      60.000000       1.000000  ...       0.000000       0.000000       0.000000       0.000000       0.000000
max    1.364800e+09     125.700000       6.800000      60.120000       1.000000  ...     121.000000      65.000000       6.000000     845.000000     121.000000

[8 rows x 12 columns]
```

Figure 4.2: AMPds/DWE web service result



```
DWE
        TS      V
1333263600 117.0
1333263660 116.9
1333263720 117.2
1333263780 117.9
1333263840 117.9
1333263900 117.7
1333263960 117.5
1333264020 117.3
1333264080 117.4
1333264140 117.3
```

Figure 4.3: AMPds/DWE/V
web service result



```
DWE
        TS      V   I
1333263600 117.0 0.0
1333263660 116.9 0.0
1333263720 117.2 0.0
1333263780 117.9 0.0
1333263840 117.9 0.0
1333263900 117.7 0.0
1333263960 117.5 0.0
1333264020 117.3 0.0
1333264080 117.4 0.0
1333264140 117.3 0.0
```

Figure 4.4: AMPds/DWE/V+I
web service result

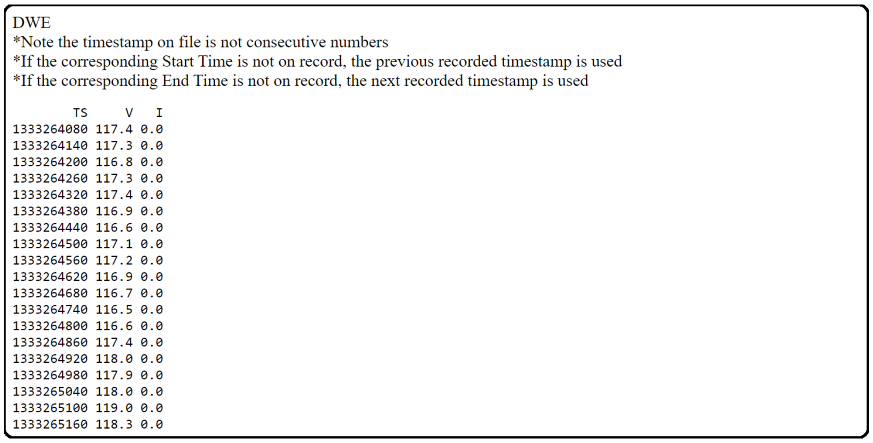
Figure 4.2 shows the user what file was selected. It would also display the minimum timestamp, maximum timestamp, and number of entries for time-series data for the selected file. The minimum and maximum timestamp helped the user determine the timestamp that they wanted to enter in the next step. Statistical information for the entire file was also shown, which included the following: count, mean, standard deviation, minimum, maximum, and the 25th, 50th, and 75th percentiles. The statistical information provided would give the user a basic overview of the statistical information about the data.

Further selection(s) of the columns TS, V, I, f, DPF, APF, P, Pt, Q, Qt, S, and St could be managed by inputting the requested column into the URL. A demonstration of the requesting column V from the selected file was shown as the URL below:
http://localhost:8000/AMPds/DWE/V. The column TS was the timestamp for the data, so it would always be included and would always be shown in the web service result when requesting information regarding other columns. Figure 4.3 shows the requested columns. When no timestamp was requested, the first 10 lines of the data starting from the smallest timestamp of the columns were displayed.

To return multiple columns such as V and I, the requested URL would be:
http://localhost:8000/AMPds/DWE/V+I and the returned information would be shown in Figure 4.4. As usual, if the user did not specify the timestamp, the first 10 lines of the requested columns would be displayed starting from the minimum timestamp.

To select the columns V and I with the specific timestamp, the characters @ and ∼ were used. The start timestamp was set to 1333264100 and the end timestamp was set to be 1333265110. The input URL was:
http://localhost:8000/AMPds/DWE/V+I@1333264100∼1333265110 as shown in Figure 4.5.

The requested start timestamp was 1333264100. Looking at Figure 4.4, we noticed that 1333264100 was between the recorded timestamp 1333264080 and 1333264140 because the recorded timestamps were not consecutive numbers. In order not to miss any information from the timestamp that the user might want to use, if the corresponding start time was



```
DWE
*Note the timestamp on file is not consecutive numbers
*If the corresponding Start Time is not on record, the previous recorded timestamp is used
*If the corresponding End Time is not on record, the next recorded timestamp is used

        TS     V   I
1333264080 117.4 0.0
1333264140 117.3 0.0
1333264200 116.8 0.0
1333264260 117.3 0.0
1333264320 117.4 0.0
1333264380 116.9 0.0
1333264440 116.6 0.0
1333264500 117.1 0.0
1333264560 117.2 0.0
1333264620 116.9 0.0
1333264680 116.7 0.0
1333264740 116.5 0.0
1333264800 116.6 0.0
1333264860 117.4 0.0
1333264920 118.0 0.0
1333264980 117.9 0.0
1333265040 118.0 0.0
1333265100 119.0 0.0
1333265160 118.3 0.0
```

Figure 4.5: AMPds/DWE/V+I@1333264100~1333265110 web service result

not on record, then the previous recorded timestamp would be used. If the corresponding end time was not recorded, then the next recorded timestamp would be used. The requested end timestamp was 1333265110. 1333265110 was not a recorded timestamp, it was between 1333265100 and 1333265160 as shown in Figure 4.5. To avoid missing any information, the latter or the next recorded timestamp was used.

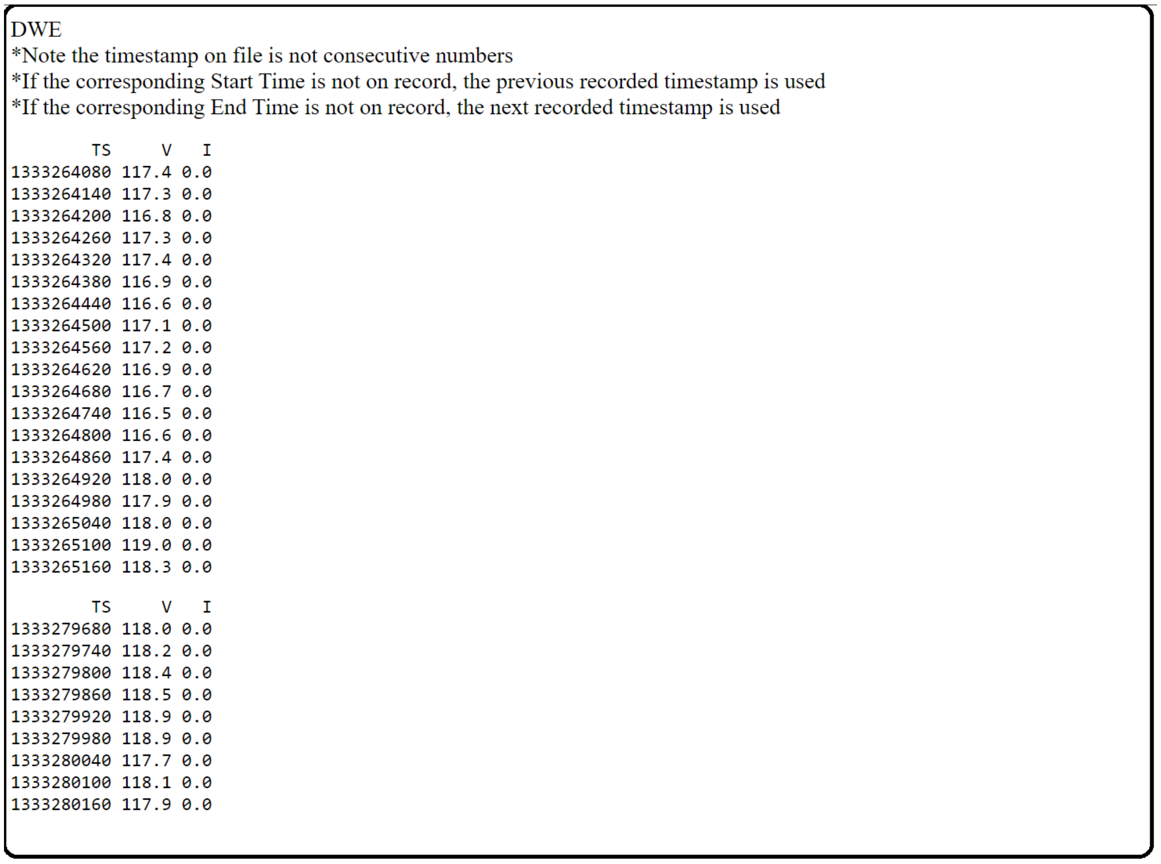
Just as we could use + for requesting more than one column, we could also use + to request more than one timestamp. The following URL requested two periods of time, each with different timestamps. Figure 4.6 illustrates the requested timestamps.

Figure 4.6 shows that there were two sections that were from two different selected timestamp periods. The columns displayed could be selected up to as many as the user needed. Figure 4.7 illustrates the web service result when three columns were requested in two timestamp periods.

To request the readings for a given sensor such as the dishwasher meter (DWE) for the Voltage measurement (V) over all the recorded timestamps, the following URL was used:
http://localhost:8000/AMPds/DWE/V@*

Instead of specifying the timestamps, the * symbol represented the entire duration of the recorded timestamps. Figure 4.8 illustrates the first few rows of the web service result when the voltage was selected. Due to the large amount of information displayed, only the first few rows and the last few rows were shown as figures here. Figure 4.9 illustrates the last few rows of the web service result. As shown in Figure 4.2, the minimum timestamp started at 1333263600, and the maximum timestamp ended at 1364799540. Both values matched with the web service results in Figure 4.8 and in Figure 4.9.

An extra feature was provided to users who wanted to use cross-validation: data could be divided into the requested chunks by extending the /number at the end of the URL. An example of the request to divide all the data into three chunks from the above example where all the readings for voltage measurement of the dishwasher meter were selected would



DWE
*Note the timestamp on file is not consecutive numbers
*If the corresponding Start Time is not on record, the previous recorded timestamp is used
*If the corresponding End Time is not on record, the next recorded timestamp is used

```
        TS     V   I
1333264080 117.4 0.0
1333264140 117.3 0.0
1333264200 116.8 0.0
1333264260 117.3 0.0
1333264320 117.4 0.0
1333264380 116.9 0.0
1333264440 116.6 0.0
1333264500 117.1 0.0
1333264560 117.2 0.0
1333264620 116.9 0.0
1333264680 116.7 0.0
1333264740 116.5 0.0
1333264800 116.6 0.0
1333264860 117.4 0.0
1333264920 118.0 0.0
1333264980 117.9 0.0
1333265040 118.0 0.0
1333265100 119.0 0.0
1333265160 118.3 0.0

        TS     V   I
1333279680 118.0 0.0
1333279740 118.2 0.0
1333279800 118.4 0.0
1333279860 118.5 0.0
1333279920 118.9 0.0
1333279980 118.9 0.0
1333280040 117.7 0.0
1333280100 118.1 0.0
1333280160 117.9 0.0
```

Figure 4.6: AMPds/DWE/V+I@1333264100∼1333265110+1333279680∼1333280160 web service result

be displayed as the URL shown below. The requested URL was:
http://localhost:8000/AMPds/DWE/V@*/3

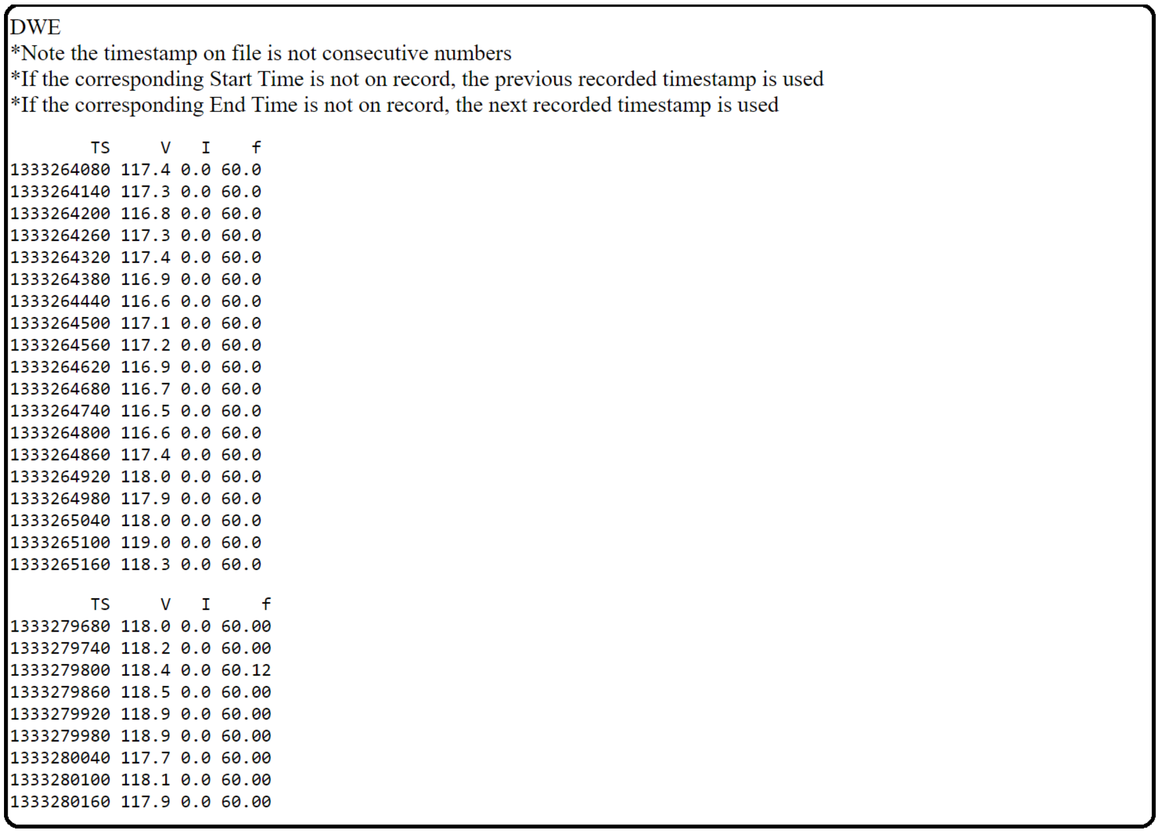
Figure 4.11 illustrates the data that were evenly divided by the number into three chunks. To make sure that each individual chunk was even in number, any reminder(s) that were indivisible by the requested number was labelled as None.

To request the time exclusion of certain timestamps, double-at @@ was used. The requested URL was shown as follows:
http://localhost:8000/AMPds/DWE/V@@1333263780∼1333263960

Figure 4.10 illustrates the web service result after the dedicated timestamps were removed. Figure 4.12 shows the project flowchart. The project flowchart provided an overview of the process described above.

4.4 Lessons Learnt

Throughout the process of creating the prototype, there were many difficulties and limitations in the implementation of the PID system with test datasets. This section talks about


DWE
*Note the timestamp on file is not consecutive numbers
*If the corresponding Start Time is not on record, the previous recorded timestamp is used
*If the corresponding End Time is not on record, the next recorded timestamp is used

```
        TS     V   I    f
1333264080 117.4 0.0 60.0
1333264140 117.3 0.0 60.0
1333264200 116.8 0.0 60.0
1333264260 117.3 0.0 60.0
1333264320 117.4 0.0 60.0
1333264380 116.9 0.0 60.0
1333264440 116.6 0.0 60.0
1333264500 117.1 0.0 60.0
1333264560 117.2 0.0 60.0
1333264620 116.9 0.0 60.0
1333264680 116.7 0.0 60.0
1333264740 116.5 0.0 60.0
1333264800 116.6 0.0 60.0
1333264860 117.4 0.0 60.0
1333264920 118.0 0.0 60.0
1333264980 117.9 0.0 60.0
1333265040 118.0 0.0 60.0
1333265100 119.0 0.0 60.0
1333265160 118.3 0.0 60.0

        TS     V   I     f
1333279680 118.0 0.0 60.00
1333279740 118.2 0.0 60.00
1333279800 118.4 0.0 60.12
1333279860 118.5 0.0 60.00
1333279920 118.9 0.0 60.00
1333279980 118.9 0.0 60.00
1333280040 117.7 0.0 60.00
1333280100 118.1 0.0 60.00
1333280160 117.9 0.0 60.00
```

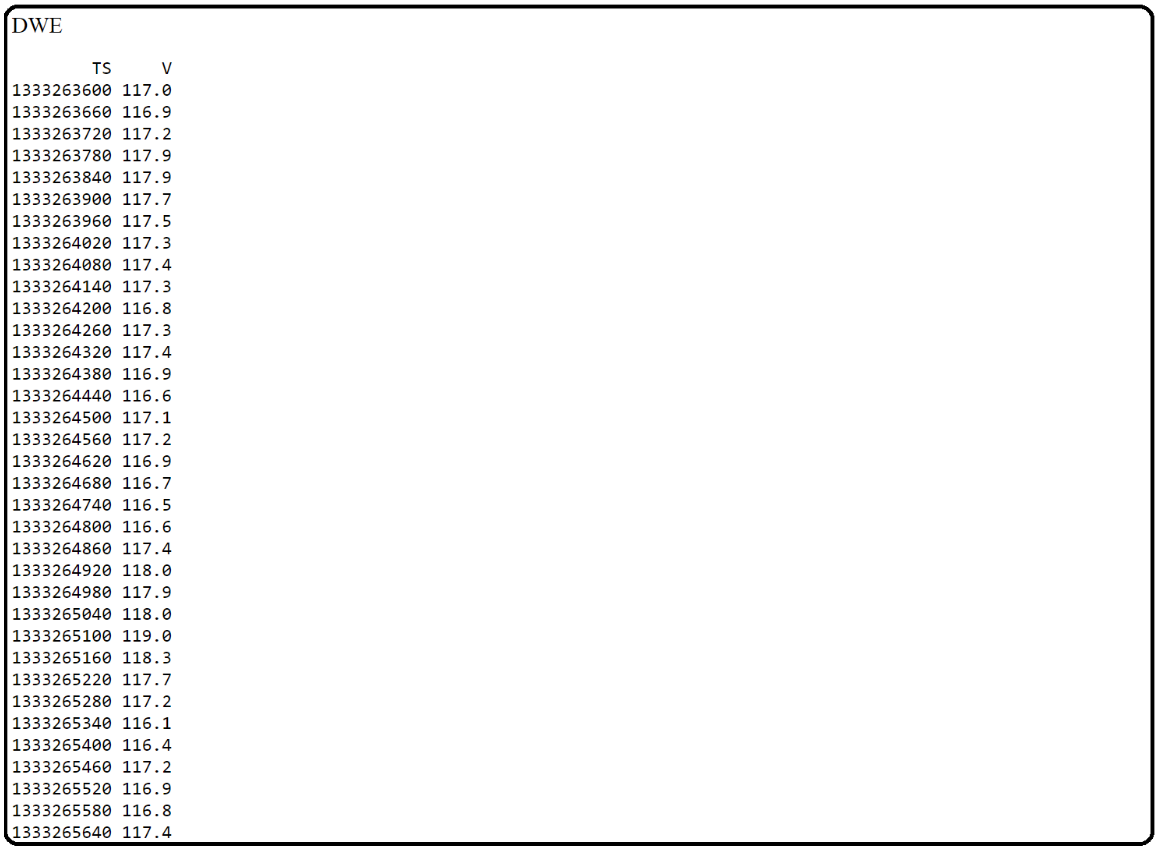
Figure 4.7: AMPds/DWE/V+I+f@1333264100∼1333265110+1333279680∼1333280160 web service result

the problems that I ran into while working on the prototype as well as some of the design decisions that I had to make to overcome these problems.

The Block diagram of a Federated Metadata Registry, as shown in Figure 2.1, is a proposed data persistence architecture that detects the dataset and categorises it into the corresponding registries. However, every dataset is different in the hierarchy and registry structure in how the data is organised. To minimize and eliminate the omission of critical useful data in the extraction process, it is critical that the system understands the structure of the dataset before initial processing.

The prototype proposed in the previous section is designed based on the structure that the AMPds dataset holds. The prototype uses the URL request format to extract information. The URL sequence follows the hierarchy structure of the dataset, which starts from the selected dataset, is followed by the sensors within, and is followed by the measurements.

The hierarchy structure also has its limitations. The hierarchy structure follows a top-down approach. Each different hierarchy acts like a folder that has subfolders within it. If the user does not know the content or the column structure of the data they are accessing, then in order to discover what subfolders are available, they have to be in the current folder. The user would only be able to see the subfolders that this current folder contains,



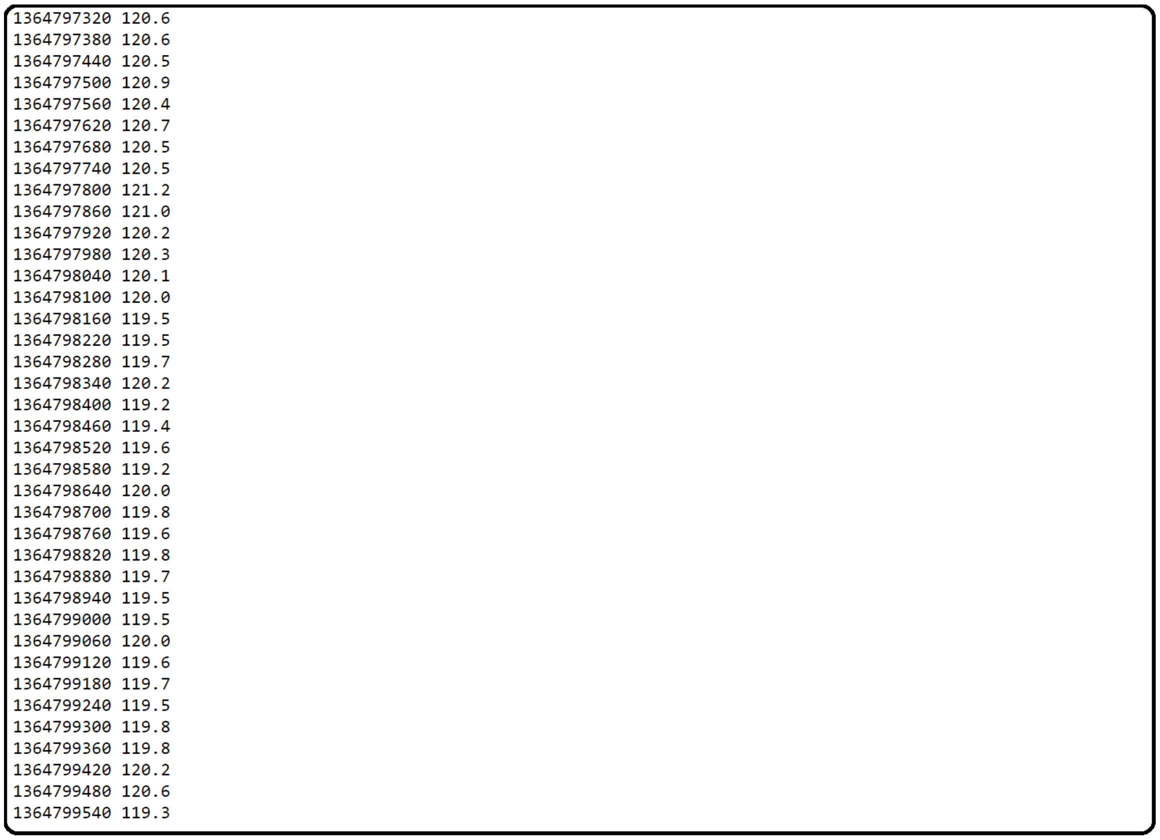
DWE

```
        TS     V
1333263600 117.0
1333263660 116.9
1333263720 117.2
1333263780 117.9
1333263840 117.9
1333263900 117.7
1333263960 117.5
1333264020 117.3
1333264080 117.4
1333264140 117.3
1333264200 116.8
1333264260 117.3
1333264320 117.4
1333264380 116.9
1333264440 116.6
1333264500 117.1
1333264560 117.2
1333264620 116.9
1333264680 116.7
1333264740 116.5
1333264800 116.6
1333264860 117.4
1333264920 118.0
1333264980 117.9
1333265040 118.0
1333265100 119.0
1333265160 118.3
1333265220 117.7
1333265280 117.2
1333265340 116.1
1333265400 116.4
1333265460 117.2
1333265520 116.9
1333265580 116.8
1333265640 117.4
```

Figure 4.8: AMPds/DWE/V@* web service result first few rows

but any content that the subfolders contain would be a mystery. The problem with this type of hierarchy structure is that if the system does not know the structure beforehand, then it would be impossible to extract the information. Therefore, knowing the column names before the extraction is essential for getting the information via URLs. When encountering this limitation, knowing the structure of the data is vital. The prototype needs to know how many hierarchical levels there are and what type of data is contained in each level to make the appropriate assessment. I eliminated the hierarchical view limitation by creating a prototype that reveals the content of the next level as the user moves into the current level.

When the user has selected the dataset, the system will return all the potential options that could be selected next. To better illustrate the approach, I will use AMPds as an example. Once the dataset AMPds is selected, the web page will return the files that are within the dataset, such as B1E.csv, B2E.csv, BME.csv, CDE.csv, CWE.csv, DNE.csv, DWE.csv, EBE.csv, EQE.csv, FGE.csv, FRE.csv, GRE.csv, HPE.csv, HTE.csv, OFE.csv, OUE.csv, TVE.csv, UTE.csv, WHE.csv, WOE.csv. Suppose that the user selects the WHE.csv file, the web page will give out the name of the columns that this file contains: TS, V, I, f, DPF, APF, P, Pt, Q, Qt, S, St. All the possible options are revealed to the users after making the



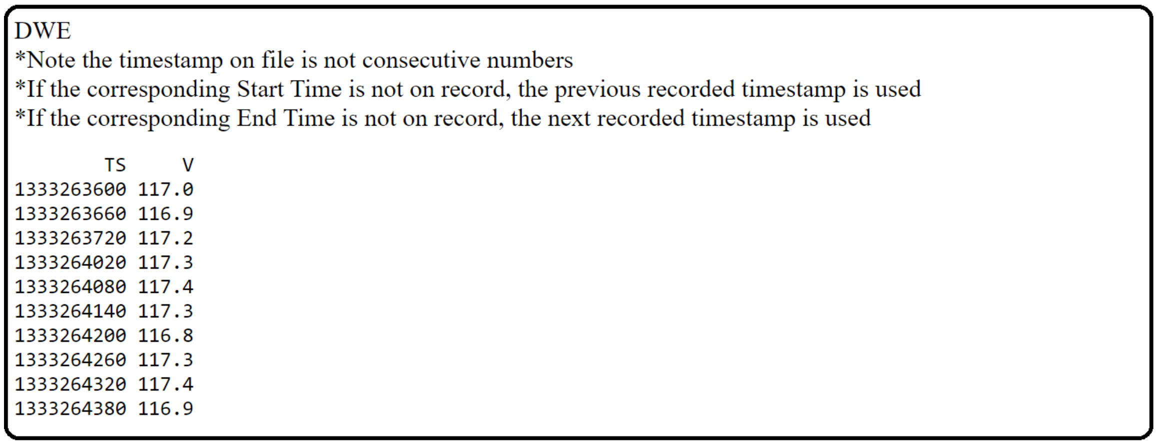
```
1364797320 120.6
1364797380 120.6
1364797440 120.5
1364797500 120.9
1364797560 120.4
1364797620 120.7
1364797680 120.5
1364797740 120.5
1364797800 121.2
1364797860 121.0
1364797920 120.2
1364797980 120.3
1364798040 120.1
1364798100 120.0
1364798160 119.5
1364798220 119.5
1364798280 119.7
1364798340 120.2
1364798400 119.2
1364798460 119.4
1364798520 119.6
1364798580 119.2
1364798640 120.0
1364798700 119.8
1364798760 119.6
1364798820 119.8
1364798880 119.7
1364798940 119.5
1364799000 119.5
1364799060 120.0
1364799120 119.6
1364799180 119.7
1364799240 119.5
1364799300 119.8
1364799360 119.8
1364799420 120.2
1364799480 120.6
1364799540 119.3
```

Figure 4.9: AMPds/DWE/V@* web service result last few rows

choice. One of the advantages of using this type of prototype is that users do not need to know the structure beforehand. Most users are not familiar with the hierarchy levels of the dataset. With this approach, they can still extract the information they need. The limitation of using this type of approach is that the user would have a very limited vision of the data. They would know the information of the current hierarchy and the options of the next hierarchy level, but not so much of the overall data structure. In addition to the limitation mentioned above, the display of the information extracted from the current level combined with the options of choices for the next level can make the layout very confusing to the users. Too much information displayed will lose the main focus of the purpose of creating the prototype, which is to extract the information given the dataset, sensor, measurements, and timestamps.

Therefore, a change of approach is applied to the prototype to emphasise the information extracted. The approach also requires the users to have a basic understanding of the dataset hierarchy levels.

Figure 2.1 proposes a system that can detect the data formats and output the corresponding registry types. This will be essential to incorporate with our model to overcome


DWE
*Note the timestamp on file is not consecutive numbers
*If the corresponding Start Time is not on record, the previous recorded timestamp is used
*If the corresponding End Time is not on record, the next recorded timestamp is used

```
        TS      V
1333263600 117.0
1333263660 116.9
1333263720 117.2
1333264020 117.3
1333264080 117.4
1333264140 117.3
1333264200 116.8
1333264260 117.3
1333264320 117.4
1333264380 116.9
```

Figure 4.10: AMPds/DWE/V@@1333263780∼1333263960 web service result

the limitation of obtaining the information about the dataset before processing in the future work.

In the developed prototype, the symbol + is used multiple times throughout the prototype, it can be used when requesting multiple measurements at multiple timestamps. In the initial design of the prototype, the + symbol is also considered at the sensor level. For example, the URL to request two different sensor measurements at the selected timestamp would be: DWE+AWE/V+I@1333264100∼1333265110. However, not all sensors would have the same recorded timestamps. If this function is implemented for sensors, there would be some void or empty information returned. The entire dataset has a large number of lines of content. If it were to display all of the content at once when requesting the information, it would take a long time to load. Therefore, only the first 10 lines are shown as a result.

Both AMPds and REFIT are time-series datasets on which our prototype is going to be tested. The REFIT dataset is organised differently from the AMPds. Both contain time-series data, so each row is a periodic reading associated with a timestamp. However, the REFIT dataset has twenty houses with one measurement, which is power, and the AMPds dataset has only one house with eleven measurements. The REFIT data is organised as the following: house(file)/appliance(column)/measurement value(cell) and AMPds is organized as: appliance(file)/measurement(column)/value(cell). The REFIT Electrical Load Measurements dataset shares a similar dataset structure as the AMPds dataset. The REFIT dataset has 20 files each representing a different house that records a number of appliances at the applied timestamps. Therefore, extracting the necessary information using the URL format would also apply to the REFIT dataset because the underlying hierarchy structure for REFIT also follows and can be represented as: *file/column/cell*. The REFIT dataset contains electrical consumption data in watts for 20 households sampled at 8-second intervals [138]. The version of the REFIT dataset that is used for testing with the built prototype has been processed and cleaned. Duplicate timestamps have been merged, and all readings above the rated limit of the sensor have been set to 0 [138]. REFIT dataset shares a similar dataset structure as the AMPds dataset. Therefore, extracting the dataset for REFIT using the built prototype would also work well.

Both datasets tested with the prototype are the cleaned-up versions with the duplicates merged and the unreasonable ratings removed. If using other datasets for analysis that do not have the same processing for data duplicates and removal of over-the-limit readings, quality of the data extraction would be affected.

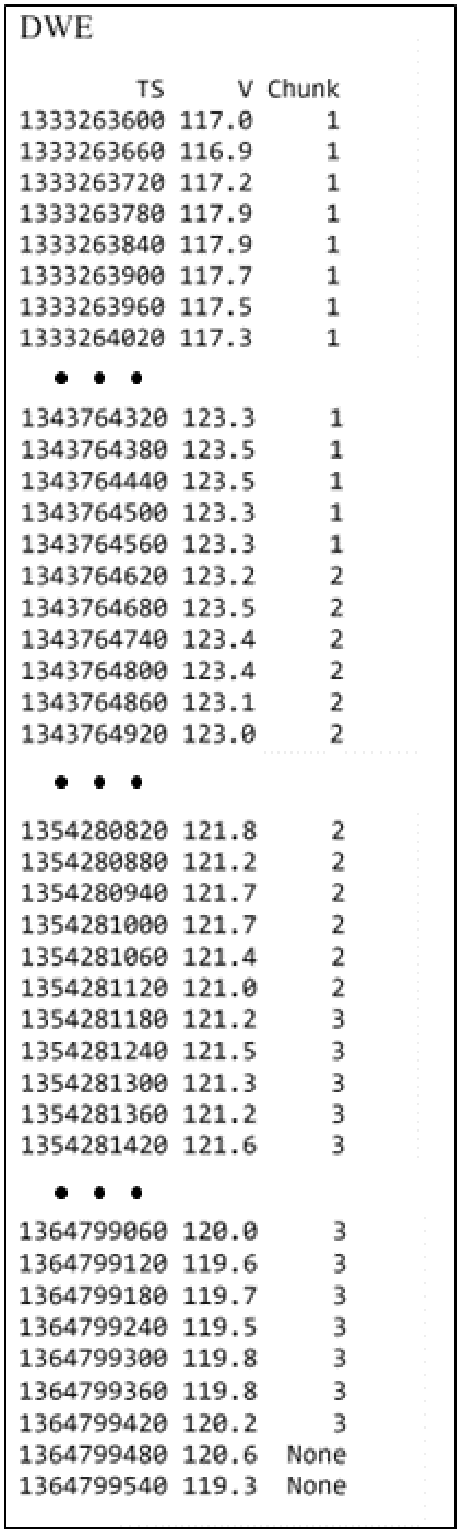
If a dataset with a different hierarchy structure were to be used for testing with the existing prototype, data parsing and data formatting would have to be done. To use this built prototype, time-series data with a timestamp is needed. The timestamp is used as the primary key to extract the information associated with it. Both the REFIT and AMPds test datasets contain CSV files for extraction. If other types of data formats are used, data parsing is needed to convert data from one format to a CSV file. Data parsing also helps convert the data files into a more readable format, which could make the files more human-readable friendly.

The prototype works with flat files such as *comma-separated value* (CSV) files and uses web formats such as URLs to display the extracted information. The same extraction could also be done using the structured query language (SQL). SQL is a programming language that can be used to process information stored in flat files. Using the built prototype for data extraction to extract information provides benefits for people who do not necessarily have the skill to code. The process is done in a fast and secure way. The extracted information can be placed on the same page side by side for ease of comparison. The same procedure can also be performed via an SQL language that requires multiple lines of code. Coding in the SQL language could be complicated and error-prone if the code is not implemented properly. However, typing the request in a URL format can be much easier and much faster to complete. As a result, extraction using the URL format lowers the barrier for people who have a very limited background in coding, but are interested in extracting from a dataset.

The existing issues of working with different dataset structures can be easily handled if a database is built based on the Block diagram of the Federated Metadata Registry described in Figure 2.1. The proposed Federated Metadata Registry promotes an easier editing process and adds restrictions to enforce data integrity. The proposed Federated Metadata Registry maintains the data integrity of the data to achieve a high level of precision and maintain outstanding quality. Some common factors that can prevent a dataset such as the Federated Metadata Registry from achieving data integrity include human error, inconsistencies between formats, collection errors, and cybersecurity privacy breaches [139]. The proposed registry allows preprocessing and categorisation of the data into corresponding classified divisions, which eliminates inconsistencies across formats. With a database like the proposed Federated Metadata Registry, it is easier to edit and add restrictions, which can reduce the chance of human errors. Finally, the database is built with high cybersecurity in mind that does not allow ease of changes on the dataset itself.

4.5 Performance Measurement

Traditionally one might wonder the performance of our proposed PID system; however, performance would be directly tied to the performance of the hardware of the server this system runs on — both server specifications and running load. This is also the case for measuring server versus just running similar code on the local client machine. The main purpose of our proposed PID system is to try to aid in the solution of reproducible science. We are trying to eliminate the need for a researcher to find the dataset used and write code to hopefully manipulate/wrangle the date in the same way the authors of the published research did. So, for us, performance is based on reducing the number of manual tasks, as demonstrated in Section 1.4. Such design principles as *single-click* and *write once, run anywhere* (WORA) [140] were used. By reducing manual tasks, we can start to eliminate any reproducible mistakes/errors and the time needed to verify the published research results. As well, this eliminates the need for every research that wants to reproduce the results of the same published to have to write their own code, hopefully without bugs that would affect their results to differ from the published ones.



DWE

```
        TS     V Chunk
1333263600 117.0     1
1333263660 116.9     1
1333263720 117.2     1
1333263780 117.9     1
1333263840 117.9     1
1333263900 117.7     1
1333263960 117.5     1
1333264020 117.3     1
  • • •
1343764320 123.3     1
1343764380 123.5     1
1343764440 123.5     1
1343764500 123.3     1
1343764560 123.3     1
1343764620 123.2     2
1343764680 123.5     2
1343764740 123.4     2
1343764800 123.4     2
1343764860 123.1     2
1343764920 123.0     2
  • • •
1354280820 121.8     2
1354280880 121.2     2
1354280940 121.7     2
1354281000 121.7     2
1354281060 121.4     2
1354281120 121.0     2
1354281180 121.2     3
1354281240 121.5     3
1354281300 121.3     3
1354281360 121.2     3
1354281420 121.6     3
  • • •
1364799060 120.0     3
1364799120 119.6     3
1364799180 119.7     3
1364799240 119.5     3
1364799300 119.8     3
1364799360 119.8     3
1364799420 120.2     3
1364799480 120.6  None
1364799540 119.3  None
```

Figure 4.11: AMPds/DWE/V@*/3 web service result

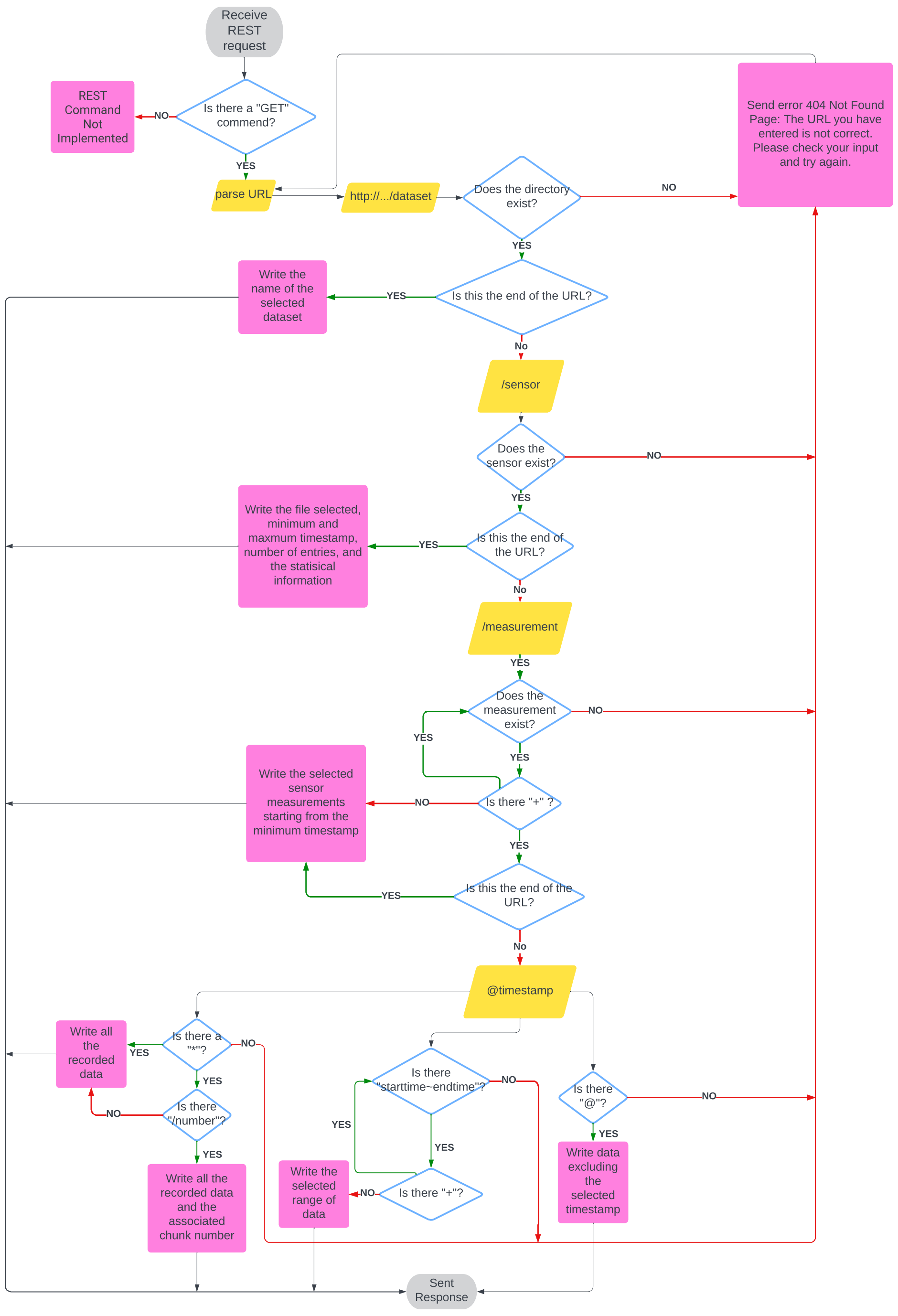

Receive REST request
Is there a "GET" commend?
NO
REST Command Not Implemented
YES
parse URL
http://.../dataset
Does the directory exist?
NO
Send error 404 Not Found Page: The URL you have entered is not correct. Please check your input and try again.
YES
Is this the end of the URL?
YES
Write the name of the selected dataset
No
/sensor
Does the sensor exist?
NO
YES
Is this the end of the URL?
YES
Write the file selected, minimum and maxmum timestamp, number of entries, and the statisical information
No
/measurement
YES
Does the measurement exist?
NO
YES
YES
Is there "+" ?
NO
Write the selected sensor measurements starting from the minimum timestamp
YES
Is this the end of the URL?
YES
No
@timestamp
Is there a "*"?
YES
Write all the recorded data
NO
YES
Is there "/number"?
NO
YES
Write all the recorded data and the associated chunk number
Is there "starttime~endtime"?
NO
YES
YES
Is there "+"?
NO
Write the selected range of data
Is there "@"?
NO
YES
Write data excluding the selected timestamp
Sent Response

Figure 4.12: Project Flowchart

Chapter 5

Conclusion

Reproducible science benefits the research community because it provides users credibility of the data, which promotes accelerated progress in methodological research and innovation. Data, documents, and results are presented and stored in a reliable location. However, there are factors that prevent users from reaching reproducibility: complexity, technological change, human error, and concerns about intellectual property rights.

Nonetheless, our demonstration (Contributes 1 and 2, as per Section 1.4) of how to use PIDs helps to eliminate the difficulties of reaching reproducibility. PID provides a high standard for finding and storing qualified data. It is a long-lasting reference to the digital resource and allows specific data to be queried and accessed from a request originating from outside the system. PID prevents unintentional alteration or corruption during the processing stage.

Our prototype (Contribution 3, as per Section 1.4) is designed specifically for reproducible science using time-series data. It is proposed and built based on the IEEE Standards Association P2957 Data Governance and Metadata Management Working Group (BDGMMWG). The database follows the block diagram (Fig. 2.1) of a BDGMMWG federated metadata registry. The datasets used for testing contain time-series data which is organized using the Timeline Index data structure and uses a timestamp as its primary key.

5.1 Future Work

ARK provides flexible persistent access to specific data from a dataset. The data provided can be used to reach reproducibility for research and scientific results. Catalog Metadata Registry, as mentioned in Chapter 2, can be summarized as a standard metadata model in which each unique data element can be represented in the following components: core, domain, relation, model, dictionary, and schema. Also mentioned in Chapter 2, Types Metadata Registry can identify, define, and register data from a dataset into different data types which allows different operations on similar datatypes. For example, numeric data types can have mathmatical operations applied to them; however, text datatypes cannot.

The existing prototype can extract data from one dataset using requests in the form of a URL. However, many datasets share similar categorizations as identified by the Catalog Metadata Registry and Types Metadata Registry. Data from related datasets can be used for comparisons. One question that brings up the attention is: *Can we link more than one dataset together in a PID?*

Future work of the project involves the linkage of different datasets containing time-series data as an additional feature. Suppose we are given the AMPds data with the following ARK PID:

F1: https://n2t.net/ark:/57460/AMPds/DWE/V@13332∼13400

Given the information extracted from the AMPds dataset, we are interested to know if there are other datasets that can be linked to PID F1. The requesting PID can be written as the following ARK PID:

F2: https://n2t.net/ark:/57460/AMPds?links

The requesting PID F2 would list out all the other datasets that can be linked to the AMPds dataset in PID F1. Suppose that the results are given back in PID F3 and PID F4:

F3: https://n2t.net/ark:/62700/Weather/YVR, (TS, DateTime)

F4: https://n2t.net/ark:/432000/EnvCan/WaterUsage, (TS, sample_date)

PID F3 and PID F4 show that the data from Weather/YVR and the data from EnvCan/WaterUsage can be linked to the AMPds dataset. The column TS from the AMPds dataset in PID F1 can link to the column DataTime from the Weather/YVR in PID F3. Similarly, the column TS from the AMPds dataset in F1 can link to the column sample_date from the EnvCan/WaterUsage in F4. The information inside the brackets (TS, DateTime) and (TS, sample_date) indicates the columns that can be linked.

F5: https://n2t.net/ark:/57460/AMPds/DWE/V@*?links/62700/Weather/YVR/temp

PID F5 shows the linkage between the AMPds dataset and the Weather dataset. The user is interested in getting the temperature information from the Weather/YVR dataset and linking all matching timestamps with the DWE sensor data in AMPds.

The prototype in this thesis works primarily with time-series data. With a vast number of different datatypes in existence, we plan to explore and expand the implementation of the prototype for other types, other than time-series data, as our next step.

Bibliography

[1] Microsoft, “Registry Data Types.” microsoft.com. https://learn.microsoft.com/en-us/windows/win32/shell/hkey-type.

[2] Microsoft, “Registry value types.” microsoft.com. https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types.

[3] PURL Administration, “PURL help.” purl.archive.org. https://purl.archive.org/help.

[4] J. F. Kurose and K. W. Ross, *Chapter 1 Computer Networks and the Internet; Chapter 2 Application Layer*. Pearson Education, 2020.

[5] L. Richardson, M. Amundsen, and S. Ruby. O’Reilly, Sept 2013.

[6] D. Quick and K.-K. R. Choo, “Impacts of increasing volume of digital forensic data: A survey and future research challenges,” *Digital Investigation*, vol. 11, no. 4, pp. 273–294, 2014.

[7] K. Braunschweig, J. Eberius, M. Thiele, and W. Lehner, “The state of open data,” *Limits of current open data platforms*, vol. 1, pp. 72–72, 2012.

[8] W. Chang and et al., “IEEE IC Big Data Governance and Metadata Management: Standards Roadmap,” *IEEE SA Industry Connections*, p. 62, 2020.

[9] L. Lannom, D. Broeder, and G. Manepalli, “Rda data type registries working group output,” 2015.

[10] ORCiD, “Structure of the ORCID Identifier.” orcid.org. https://support.orcid.org/hc/en-us/articles/360006897674-Structure-of-the-ORCID-Identifier.

[11] Moody Medical Library, “What is a DOI number.” library.utmb.edu. https://guides.utmb.edu/DOI.

[12] University of Illinois Chicago, “What is a DOI and how do I use them in citations?.” library.uic.edu. https://ask.library.uic.edu/faq/345899.

[13] Research Organization Registry (ROR), “What is a ROR identifier?.” ror.org. https://ror.org/about/faqs/#what-is-a-ror-identifier.

[14] ARK Alliance, “About ARKs.” arks.org. https://arks.org/about/.

[15] isni, “Linked Data.” isni.org. https://isni.org/page/linked-data/.

[16] Handle.Net® Registry, "HDL.NET® Information Services." handle.net. `https://www.handle.net/index.html`.

[17] P. Saint-Andre and D. J. C. Klensin, "RFC 8141: Uniform Resource Names (URNs)." datatracker.ietf.org. `https://datatracker.ietf.org/doc/html/rfc8141`.

[18] J. F. Kurose and K. W. Ross, *2.2.1 Overview of HTTP*, p. 96. Pearson, July 2020.

[19] J. F. Kurose and K. W. Ross, *2.2.1 Overview of HTTP*, p. 96–97. Pearson, July 2020.

[20] J. F. Kurose and K. W. Ross, *2.2.3 HTTP Message Format*, p. 101–102. Pearson, July 2020.

[21] J. F. Kurose and K. W. Ross, *2.2.3 HTTP Message Format*, p. 103–104. Pearson, July 2020.

[22] H. Mann, "REST APIs Explained - 4 Components." mannhowie.com. `https://mannhowie.com/rest-api`.

[23] J. A. Kunze and E. Bermès, "The ARK Identifier Scheme," Internet-Draft draft-kunze-ark-38, Internet Engineering Task Force, Nov. 2023. Work in Progress.

[24] J. M. Alston and J. A. Rick, "A beginner's guide to conducting reproducible research," *Bulletin of the Ecological Society of America*, vol. 102, no. 2, pp. 1–14, 2021.

[25] C. Gross, "Scientific misconduct," *Annual review of psychology*, vol. 67, 2016.

[26] W. J. Broad, "Harvard delays in reporting fraud: Six months passed before officials made public a confessed case of fraud, and then only after nih questioned suspicious data," *Science*, vol. 215, no. 4532, pp. 478–482, 1982.

[27] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[28] National Library of Medicine, "Reproducibility and Replicability in Science." ncbi.nlm.nih.gov. `https://www.ncbi.nlm.nih.gov/books/NBK547546/`.

[29] IEEE Standards Association, "BIG DATA GOVERNANCE AND META-DATA MANAGEMENT: STANDARDS ROADMAP." standards.ieee.org `https://standards.ieee.org/wp-content/uploads/import/governance/iccom/bdgmm-standards-roadmap-2020.pdf`.

[30] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," *Procedia Computer Science*, vol. 161, pp. 466–474, 2019.

[31] University of Illinois at Chicago, "Research Reproducibility." library.uic.edu. `https://researchguides.uic.edu/reproducibility`.

[32] R. D. Peng, "Reproducible research in computational science," *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011.

[33] J. Heppner, "How to Better Understand Reproducibility in Science." blog.orvium.io. `https://blog.orvium.io/reproducible-science/`.

[34] S. Sirisilla, "Top 5 Factors Affecting Reproducibility in Research." enago.com/academy. `https://www.enago.com/academy/top-5-factors-affecting-reproducibility-in-scientific-research/`.

[35] P. Diaba-Nuhoho and M. Amponsah-Offeh, "Reproducibility and research integrity: the role of scientists and institutions," *BMC Research Notes*, vol. 14, no. 1, pp. 1–4, 2021.

[36] C. J. Playford, V. Gayle, R. Connelly, and A. J. Gray, "Administrative social science data: The challenge of reproducible research," *Big Data & Society*, vol. 3, no. 2, p. 2053951716684143, 2016.

[37] W. T. M. Tu and S. Makonin, "Designing pids for reproducible science using time-series data," *arXiv preprint arXiv:2209.10475*, 2022.

[38] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[39] M. Tahmassebpour, "A new method for time-series big data effective storage," *Ieee Access*, vol. 5, pp. 10694–10699, 2017.

[40] IEEE Standards Association, "IEEE Big Data Governance and Metadata Management (2957)." sagroups.ieee.org `hhttps://sagroups.ieee.org/2957/`.

[41] A. Lausch, A. Schmidt, and L. Tischendorf, "Data mining and linked open data–new perspectives for data analysis in environmental research," *Ecological Modelling*, vol. 295, pp. 5–17, 2015.

[42] O. J. Reichman, M. B. Jones, and M. P. Schildhauer, "Challenges and opportunities of open data in ecology," *Science*, vol. 331, no. 6018, pp. 703–705, 2011.

[43] V. Gligorijević and N. Pržulj, "Methods for biological data integration: perspectives and challenges," *Journal of the Royal Society Interface*, vol. 12, no. 112, p. 20150571, 2015.

[44] P. Ziegler and K. R. Dittrich, "Data integration—problems, approaches, and perspectives," in *Conceptual modelling in information systems engineering*, pp. 39–58, Springer, 2007.

[45] B. Li and L. Zhang, "Distributed spatial catalog service on the corba object bus," *GeoInformatica*, vol. 4, pp. 253–269, 2000.

[46] S. C. Guptill, "Metadata and data catalogues," *Geographical information systems*, vol. 2, pp. 677–692, 1999.

[47] S. M. Khan, X. Liu, S. Nath, E. Korot, L. Faes, S. K. Wagner, P. A. Keane, N. J. Sebire, M. J. Burton, and A. K. Denniston, "A global review of publicly available datasets for ophthalmological imaging: barriers to access, usability, and generalisability," *The Lancet Digital Health*, vol. 3, no. 1, pp. e51–e66, 2021.

[48] A. A. Sinaci and G. B. L. Erturkmen, "A federated semantic metadata registry framework for enabling interoperability across clinical research and care domains," *Journal of biomedical informatics*, vol. 46, no. 5, pp. 784–794, 2013.

[49] Y. Bai, L. Di, A. Chen, Y. Liu, and Y. Wei, "Towards a geospatial catalogue federation service," *Photogrammetric Engineering & Remote Sensing*, vol. 73, no. 6, pp. 699–708, 2007.

[50] M. P. Robillard, E. Bodden, D. Kawrykow, M. Mezini, and T. Ratchford, "Automated api property inference techniques," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 613–637, 2012.

[51] Atlan, "Federated Data Catalog: When should you go for one?." atlan.com. `https://atlan.com/federated-data-catalog/`.

[52] Open Geospatial Consortium, "Catalogue Service." ogc.org. `https://www.ogc.org/standard/cat/`.

[53] magda, "A federated catalog for all of your data." magda.io. `https://magda.io/`.

[54] J. Ison, K. Rapacki, H. Ménager, M. Kalaš, E. Rydza, P. Chmura, C. Anthon, N. Beard, K. Berka, D. Bolser, *et al.*, "Tools and data services registry: a community effort to document bioinformatics resources," *Nucleic acids research*, vol. 44, no. D1, pp. D38–D47, 2016.

[55] M. N. Levine and J. A. Julian, "Registries that show efficacy: good, but not good enough." pubmed.ncbi.nlm.nih.gov. `https://pubmed.ncbi.nlm.nih.gov/18854560/`.

[56] C. Nelson, "Use of metadata registries for searching for statistical data," in *Proceedings 14th International Conference on Scientific and Statistical Database Management*, pp. 232–235, IEEE, 2002.

[57] C. Blanchi and J. Petrone, "Distributed interoperable metadata registry," *D-Lib Magazine*, vol. 7, no. 12, pp. 1082–9873, 2001.

[58] B. E. Bargmeyer and D. W. Gillman, "Metadata standards and metadata registries: An overview," in *International Conference on Establishment Surveys II, Buffalo, NY*, 2000.

[59] J.-h. Li, J.-x. Gao, J.-n. Dong, W. Wu, and Y.-f. Hou, "A metadata registry for metadata interoperability," *Data Science Journal*, vol. 6, pp. S379–S384, 2007.

[60] P. Zhao, A. Chen, Y. Liu, L. Di, W. Yang, and P. Li, "Grid metadata catalog service-based ogc web registry service," in *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pp. 22–30, 2004.

[61] J. B. Bowen, "Moving library metadata toward linked data: Opportunities provided by the extensible catalog," in *International Conference on Dublin Core and Metadata Applications*, pp. 44–59, 2010.

[62] University of North Carolina at Chapel Hill, "Metadata for Data Management: A Tutorial: Basic Elements." library.unc.edu. `https://guides.lib.unc.edu/metadata/basic-elements`.

[63] Komprise, “Data Management Glossary.” komprise.com. https://www.komprise.com/glossary_terms/metadata/#:~:text=Some%20examples%20of%20basic%20metadata,the%20form%20of%20meta%20tags.

[64] M. Cawsey, “What is a Data Domain?.” stibosystems.com. https://www.stibosystems.com/blog/what-is-a-data-domain.

[65] J. Greenberg, “Understanding metadata and metadata schemes,” *Cataloging & classification quarterly*, vol. 40, no. 3-4, pp. 17–36, 2005.

[66] University of North Texas, “Metadata Input Guidelines: Relation.” unt.edu. https://library.unt.edu/digital-projects-unit/metadata/fields/relation/.

[67] D. I. Hillmann, R. Marker, and C. Brady, “Metadata standards and applications,” *The Serials Librarian*, vol. 54, no. 1-2, pp. 7–21, 2008.

[68] Cambridge Semantics, “Creating a Metadata Dictionary (Preview).” cambridgesemantics.com. https://docs.cambridgesemantics.com/anzo/v5.3/userdoc/data-dictionary.htm.

[69] University of North Carolina at Chapel Hill, “Metadata for Data Management: A Tutorial: Controlled Vocabularies.” library.unc.edu. https://guides.lib.unc.edu/metadata/controlled-vocab.

[70] IBM, “Metadata modeling.” ibm.com. https://www.ibm.com/docs/en/cognos-analytics/12.0.0?topic=guidelines-metadata-modeling.

[71] E. Yang, B. Matthews, and M. Wilson, “Enhancing the core scientific metadata model to incorporate derived data,” *Future Generation Computer Systems*, vol. 29, no. 2, pp. 612–623, 2013.

[72] IBM, “Planning your project.” ibm.com. https://www.ibm.com/docs/en/cognos-analytics/12.0.0?topic=modeling-planning-your-project.

[73] University of California Santa Cruz, “Metadata Creation Process.” ucsc.edu. https://guides.library.ucsc.edu/c.php?g=618773&p=4306387.

[74] ORACLE, “Introduction to the Metadata Model.” docs.oracle.com. https://docs.oracle.com/cd/E23095_01/ACI.93/ACIDataWarehouse/html/s0106introductiontothemetadatamodel01.html.

[75] D. I. Hillmann, S. A. Sutton, J. Phipps, and R. Laundry, “A metadata registry from vocabularies up: The nsdl registry project,” *arXiv preprint cs/0605111*, 2006.

[76] UC Santa Cruz, “Dublin Core Metadata Schema.” library.ucsc.edu. https://guides.library.ucsc.edu/c.php?g=618773&p=4306386.

[77] The University of North Carolina at Chapel Hill, “Metadata for data management: A tutorial.” library.unc.edu. https://guides.lib.unc.edu/metadata.

[78] University of Toronto Scarborough, “Jackman Scholars-in-Residence The Art and Science of Museum Objects and Seeing Potential: Asking/Investigating/Exhibiting the Malcove Collection.” utsc.library.utoronto.ca. https://guides.library.utoronto.ca/jhischolars.

[79] DCMI Usage Board, "Dublin Core™ Metadata Element Set, Version 1.1: Reference Description." dublincore.org. https://www.dublincore.org/specifications/dublin-core/dces/.

[80] R. Heery and H. Wagner, "A metadata registry for the semantic web," *D-Lib Magazine*, vol. 8, no. 5, pp. 15–17, 2002.

[81] Y.-K. Ki, J.-W. Kim, and D.-K. Baik, "A traffic accident detection model using metadata registry," in *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, pp. 255–259, IEEE, 2006.

[82] K. Ivens, "Registry Data Types." itprotoday.com. https://www.itprotoday.com/windows-8/registry-data-types#close-modal.

[83] S. Nativi and L. Bigagli, "Discovery, mediation, and access services for earth observation data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 2, no. 4, pp. 233–240, 2009.

[84] C. Software, "Types of Registry Data." chemtable.com/blog. https://www.chemtable.com/blog/en/types-of-registry-data.htm.

[85] L. M. Chan and M. L. Zeng, "Metadata interoperability and standardization–a study of methodology part i," *D-Lib magazine*, vol. 12, no. 6, pp. 1082–9873, 2006.

[86] A. Dappert, A. Farquhar, R. Kotarski, and K. Hewlett, "Connecting the persistent identifier ecosystem: Building the technical and human infrastructure for open research," *Data science journal*, vol. 16, pp. 28–28, 2017.

[87] National Transportation Library, "Digital Object Identifier (DOI)." ntl.bts.gov/ntl. https://transportation.libguides.com/persistent_identifiers/doi.

[88] J. Klump and R. Huber, "20 years of persistent identifiers–which systems are here to stay?," *Data Science Journal*, vol. 16, pp. 9–9, 2017.

[89] Digital Preservation Coalition, "Persistent identifiers." dpconline.org. https://www.dpconline.org/handbook/technical-solutions-and-tools/persistent-identifiers.

[90] University of New Hampshire, "Persistent Identifiers." library.unh.edu. https://libraryguides.unh.edu/datamanagement/PersistentIdentifiers.

[91] University of Washington, "Persistent Identifiers: About Persistent Identifiers." lib.washington.edu. https://guides.lib.uw.edu/research/PID.

[92] CERN Scientific Information Service, "What are persistent identifiers?." sis.web.cern.ch. https://sis.web.cern.ch/submit-and-publish/persistent-identifiers/what-are-pids.

[93] CERN Scientific Information Service, "Why use persistent identifiers?." sis.web.cern.ch. https://sis.web.cern.ch/submit-and-publish/persistent-identifiers/why-pids.

[94] National Library of Medicine, "Persistent Unique Identifier." nnlm.gov. https://www.nnlm.gov/guides/data-glossary/persistent-unique-identifier#:~:text=A%20Persistent%20Unique%20Identifier%20(PID,objects%2C%20frequently%20a%20journal%20article.

[95] M. Gould, G. Mejias, and A. Cardoso, "PIDs and Open science: Building Community in Latin America." datacite.org. https://datacite.org/blog/pids-and-open-science-building-community-in-latin-america/.

[96] K. Garza, "Breaking a Metadata Barrier: Improving discoverability with automatic subject classification." datacite.org. https://doi.org/10.5438/vwp0-vz05.

[97] E. Young, "ORCIDat10: Celebrating 10 Years of the ORCID Galaxy." orcid.org. https://info.orcid.org/orcidat10-celebrating-10-years-of-the-orcid-galaxy/#:~:text=By%20the%20end%20of%20the,time%20staff%20among%2014%20countries.

[98] L. Haak, "ORCID Launches Registry." orcid.org. https://info.orcid.org/orcid-launches-registry/#:~:text=ORCID%2C%20was%20established%20in%202010,organization%20serving%20the%20research%20community.

[99] National Transportation Library, "Introduction to Persistent Identifiers." ntl.bts.gov/ntl. https://transportation.libguides.com/persistent_identifiers.

[100] ORCiD, "What are persistent identifiers (PIDs)." support.orcid.org. https://support.orcid.org/hc/en-us/articles/360006971013-What-are-persistent-identifiers-PIDs.

[101] doi Foundation, "Key Facts on Digital Object Identifier System." doi.org. https://www.doi.org/the-identifier/resources/factsheets/key-facts-on-digital-object-identifier-system.

[102] DataCite, "Connecting research, advancing knowledge." datacite.org. https://datacite.org/.

[103] Research Organization Registry (ROR), "Registry." ror.org. https://ror.org/registry/.

[104] Research Organization Registry (ROR), "What is ROR?." ror.org. https://ror.org/about/.

[105] ARK Alliance , "ARK overview." arks.org. https://arks.org/about/ark-overview/#:~:text=What%20ARKs%20are%20and%20why,%2C%20software%2C%20websites%2C%20etc.

[106] ARK Alliance, "ARK Alliance." arks.org. https://arks.org/.

[107] ARK Alliance, "ARK Alliance." arks.org. https://arks.org/.

[108] M. Kelly, D. Ivanovic, C. B. Rauch, J. Kunze, S. Grabus, J. Boone, P. M. Logan, and J. Greenberg, "Archival resource keys for collaborative historical ontology publication," in *Proceedings of the ICTeSSH 2021 conference*, PubPub, 2021.

[109] ARK Alliance, "ARK implementation best practices." arks.org. https://arks.org/about/best-practices/.

[110] isni, "About ISNI." isni.org. https://isni.org/.

[111] isni, "What is ISNI?." isni.org. https://isni.org/page/what-is-isni/.

[112] isni, "ISNI Origins." isni.org. https://isni.org/page/our-history/.

[113] doi Foundation, "DOI® System and the Handle System®." doi.org. https://www.doi.org/the-identifier/resources/factsheets/doi-system-and-the-handle-system.

[114] DONA Foundation, "The Handle System." dona.net. https://www.dona.net/handle-system.

[115] Handle.Net® Registry, "Payment." handle.net. https://www.handle.net/payment.html.

[116] doi Foundation, "DOI® SYSTEM AND PERSISTENT URLS (PURLS)." doi.org. https://www.doi.org/the-identifier/resources/factsheets/doi-system-and-persistent-urls.

[117] L. Stone, "Competitive Evaluation of PURLs." web.mit.edu. http://web.mit.edu/handle/www/purl-eval.html.

[118] T. Berners-Lee, "Universal resource identifiers in www: a unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web," tech. rep., 1994.

[119] H.-W. Hilse, *Implementing persistent identifiers.* Consortium of European Research Libraries, 2006.

[120] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces.* " O'Reilly Media, Inc.", 2011.

[121] A. Arcuri, "Restful api automated test case generation," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 9–20, IEEE, 2017.

[122] A. Ehsan, M. A. M. Abuhaliqa, C. Catal, and D. Mishra, "Restful api testing methodologies: Rationale, challenges, and solution directions," *Applied Sciences*, vol. 12, no. 9, p. 4369, 2022.

[123] X. Chen, Z. Ji, Y. Fan, and Y. Zhan, "Restful api architecture based on laravel framework," in *Journal of Physics: Conference Series*, vol. 910, p. 012016, IOP Publishing, 2017.

[124] P. Sanjay, *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS.* Apress, March 2017.

[125] IBM, "Handling non-JSON data in a REST API." ibm.com. https://www.ibm.com/docs/en/app-connect/12.0?topic=apis-handling-non-json-data-in-rest-api.

[126] M. Biehl, *RESTful Api Design*, vol. 3. API-University Press, 2016.

[127] J. Kunze, "ARK Identifiers FAQ." wiki.lyrasis.org. https://wiki.lyrasis.org/display/ARKs/ARK+Identifiers+FAQ.

[128] ARK Alliance, "General identifier concepts and conventions." arks.org. https://arks.org/about/identifier-concepts-and-conventions/.

[129] ARK Alliance, "Getting started: what to plan for as you implement ARKs." arks.org. https://arks.org/about/getting-started-implementing-arks/.

[130] N2T.net, "About N2T.net." n2t.net. https://n2t.net/e/about.html.

[131] ARK Alliance, "Running minters and resolvers." arks.org. https://arks.org/about/running-minters-and-resolvers/.

[132] ARK Alliance, "ARK namespaces and sub-namespaces." arks.org. https://arks.org/about/ark-namespaces/.

[133] N2T.net, "NOID: Nice Opaque IDentifier (minter and name resolver)." n2t.net. https://n2t.net/e/noid.html.

[134] n2t.net, "Registered ARK Name Assigning Authority Numbers." n2t.net. https://n2t.net/e/pub/naan_table.html.

[135] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajić, "Ampds: A public dataset for load disaggregation and eco-feedback research," in *2013 IEEE Electrical Power & Energy Conference*, pp. 1–6, 2013.

[136] S. Makonin, B. Ellert, I. V. Bajic, and F. Popowich, "Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014," *Scientific Data*, vol. 3, no. 160037, pp. 1–12, 2016.

[137] T. Preston-Werner, "TOML [Tom's Obvious Minimal Language]." toml.io. https://toml.io/en/.

[138] D. Murray, L. Stankovic, and V. Stankovic, "REFIT: Electrical Load Measurements (Cleaned)." pureportal.strath.ac.uk. https://pureportal.strath.ac.uk/en/datasets/refit-electrical-load-measurements-cleaned.

[139] C. Cote, "What is data integrity and why does it matter?." online.hbs.edu/. https://online.hbs.edu/blog/post/what-is-data-integrity.

[140] Wikipedia contributors, "Write once, run anywhere — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Write_once,_run_anywhere&oldid=1211211259, 2024.