Geospatial Data Curation Guidelines for CZMAI's Coastal and Marine Data Centre

Ocean Country Partnership Programme (OCPP)

March 2025

Recommended citation: MarFishEco Fisheries Consultants. (2025). *Geospatial Data Curation Guidelines*. Published on behalf of OCPP.

Cover photo: © Chabil Mar Vilas

Evidence Quality Assurance: This report is compliant with JNCC's Evidence Quality Assurance Policy.

Ocean Country Partnership Programme:
The Ocean Country Partnership Programme (OCPP) is a UK Government-led programme delivered under the Blue Planet Fund in Overseas Development Assistance (ODA) eligible countries. Through this programme, Cefas, JNCC and MMO will provide technical assistance to support countries to tackle marine pollution, support sustainable seafood practices and establish designated, well-managed and enforced marine protected areas.

Table of Contents

List of Acronyms

Acronym	Full Term
CODATA	Committee on Data of the International Science Council
CGIAR	Consultative Group on International Agricultural Research
CMDC	Coastal and Marine Data Centre
CRS	Coordinate Reference System
CZMAI	Coastal Zone Management Authority and Institute
EPSG	European Petroleum Survey Group (commonly used for spatial reference codes)
FAIR	Findable, Accessible, Interoperable, Reusable
GCOS	Global Climate Observing System
GIS	Geographic Information System
ICZMP	Integrated Coastal Zone Management Plan
ISO	International Organization for Standardization
KML/KMZ	Keyhole Markup Language / Keyhole Markup Language Zipped
MFE	MarFishEco Fisheries Consultants
MPA	Marine Protected Area
NA	Not Available
ND	No Data
OCPP	Ocean Country Partnership Programme
OGC	Open Geospatial Consortium
QGIS	Quantum Geographic Information System
R	R Statistical Software
SHP	Shapefile
TIFF	Tagged Image File Format
UTM	Universal Transverse Mercator
VPN	Virtual Private Network

Introduction

Effective data curation and management are essential for preserving datasets and ensuring their long-term accessibility. A standardized, systematic approach to organizing, documenting, and preserving data enhances its reliability, usability, and overall application. Well-managed data not only supports future research and informs policy development but also facilitates cross-disciplinary collaboration.

Recognizing the importance of robust data management the Joint Nature Conservation Committee (JNCC), acting as a delivery partner for the UK government-led Ocean Country Partnership Programme (OCPP), has engaged MarFishEco Fisheries Consultants (MFE) to collaborate with the Coastal Zone Management Authority and Institute (CZMAI) in Belize. This partnership aims to strengthen spatial data management for coastal and marine environments, improving data accessibility and integration to support effective decision-making. By enhancing marine science expertise, the initiative advances evidence-based policy and management, contributing to the conservation of natural resources—one of OCPP's broader objectives under the Blue Planet Programme.

To support this effort, the following guidelines establish best practices for creating and maintaining a comprehensive data archiving system for CZMAI. These guidelines align with international data management standards, including those set by the Committee on Data of the International Science Council (CODATA), the Open Geospatial Consortium (OGC), the Global Climate Observing System (GCOS), the International Organization for Standardization (ISO 19115 for geospatial metadata), and the CGIAR (Consultative Group on International Agricultural Research).

These following guidelines apply to both historical and current datasets managed by CZMAI and its partners, ensuring that data is structured, accessible, and well-documented to meet immediate operational needs while supporting long-term sustainability and interoperability.

Navigating this Document

This document provides a structured, step-by-step guide for curating and managing spatial data, ensuring that datasets are accurate, well-documented, and accessible for future use. It is primarily intended for data managers, GIS analysts, and technical staff responsible for handling, processing, and maintaining marine and coastal spatial datasets. While this guide is not specifically designed for data collectors, it still serves as a valuable resource for understanding proper data recording practices and the key elements that make datasets useful and compliant with curation standards.

This document serves as both a practical guide for immediate data curation and a forward-looking resource to support the ongoing advancement of data management in Belize.

Key Features:

- **Detailed Workflow of the Data Curation Process:** A step-by-step workflow to guide users from data acquisition to storage, ensuring consistency and compliance with best practices.
- **Geospatial Data Curation Guidelines:** Each section of the workflow links to detailed guidelines on data formatting, quality assurance, metadata management, validation, and storage.
- **Practical Examples:** Includes practical examples throughout the guidelines to emphasize points are primarily based on Integrated Coastal Zone Management Plan (ICZMP) data, along with specific instructions using Arc/QGIS for error detection and correction.
- **Recommendations for Data Security, Sharing, and User Training:** Outlines key recommendations to enhance data management and sharing protocols, along with user support. While Belize may not yet have a centralized data repository or formalized training programs, these recommendations establish a foundation for improving data security, streamlining data-sharing protocols, and offering practical use cases to help data users effectively access and apply available resources.

The Data Curation Process: From Data Collection to Preservation

Effective data curation ensures that datasets are accurate, well-documented, and readily accessible for future use. The following steps (Figure 1) outline the structured process to follow once you receive data from a collector or provider and plan to curate it. Each step plays a crucial role in maintaining data integrity, usability, and compliance with curation standards.

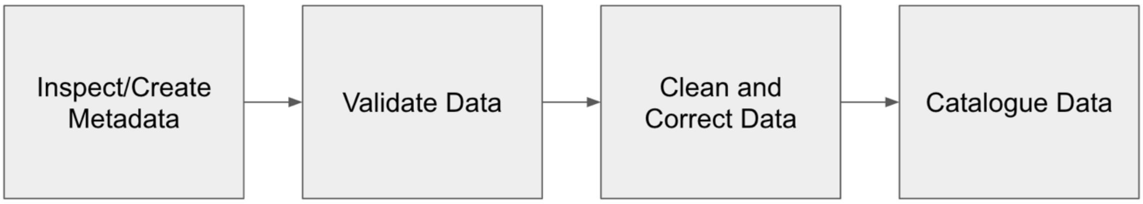


Figure 1. Steps of data curation from inspecting/creating metadata to cataloguing data.

Steps of Data Curation

1. **Inspect and/or Create Metadata**
   - Examine the metadata accompanying the dataset to ensure it is complete and accurate. Refer to the Metadata section for completeness criteria.
   - If metadata is missing or incomplete, collaborate with the data provider to address and correct gaps. Where necessary, supplement metadata using available data. Refer to Creating or Editing Metadata of a Shapefile for step-by step guidance on how to create a metadata for a shapefile in QGIS.
   - Maintain open communication with the data provider to ensure accuracy and completeness of metadata.
2. **Validate data**
   - Cross-check metadata against the actual dataset to confirm consistency.
   - Ensure data format and quality align with data curation standards. Refer to the Data Format and Quality section for detailed formatting and quality requirements.
   - Identify spatial or topological errors. Refer to the Quality Assurance and Error Checking section for detailed steps on identifying common errors.
   - Ensure that all unique features have associated spatial attributes and complete records.
   - Identify duplicate entries to enhance data reliability.
3. **Clean and Correct Data**
   - Resolve errors, inconsistencies, and anomalies identified during validation to improve data quality. Refer to Data Format and Quality and Quality Assurance and Error Checking for detailed steps on correcting common errors.
   - Document all corrections and changes made to the datafile in the metadata to ensure transparency and reproducibility.
   - If applicable, save the cleaned or corrected data as a new version-controlled file, preserving the original dataset. Document all modifications in the data file's metadata to provide transparency and traceability.

4. **Catalogue data**
    - Enter information about the data file into a data inventory catalogue, following template developed for CZMAI to ensure consistency and accessibility.
    - If applicable, provide the data collector/provider with the updated data file and how the data has been cleaned/transformed.

Geospatial Data Curation Guidelines

Data Format and Quality

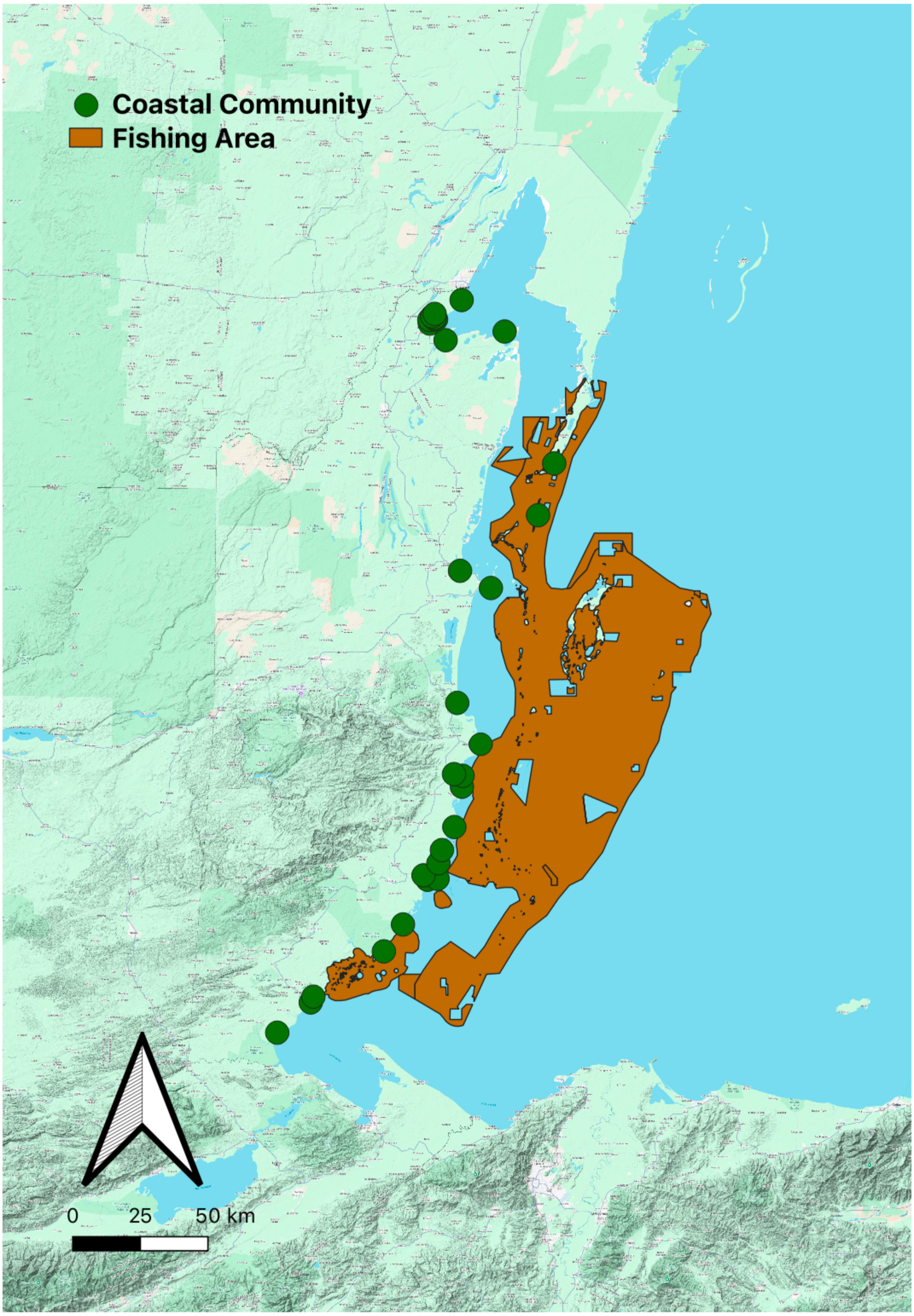
Maintaining consistent data quality and format ensures spatial datasets are accurate, interoperable, and easy to use across platforms. The following guidelines provide a framework for establishing and maintaining high-quality data standards that enhance quality and usability.

1. Spatial Data Formats

Spatial data comes in many forms, and the type of data used often depends on the specific application, such as mapping, analysis, or visualization. Using widely recognized file formats ensures that datasets can be easily shared, accessed, and used across different platforms and tools without compatibility issues.

- **Accepted File Formats**: Store geospatial data in widely recognized formats to maximize compatibility across platforms and tools. Recommended formats include:
    - **Vector Data**:
        - Shapefiles: A vector data format developed by Esri that stores geometric shapes such as points, lines, and polygons, frequently used in ArcGIS, QGIS, and R (Figure 2).
        - GeoJSON: A vector data format developed by JSON that stores geometric shapes such as points, lines, and polygons, commonly used in QGIS and web-based mapping applications.
        - KML/KMZ: XML-based formats for geographic data visualization, commonly used in Google Earth.

Figure 2. Example of two shapefiles depicting coastal communities (green points) and fishing areas (orange polygons) in Belize.



- **Raster Data**:
  - GeoTIFF: A raster data format commonly derived from satellite imagery and elevation models, widely utilized in ArcGIS, QGIS, and R (Figure 3).
  - GRID: A proprietary raster format by Esri, used in ArcGIS and QGIS.
  - NetCDF: A raster data format commonly used to represent climate and atmospheric data, frequently utilized in ArcGIS, QGIS, R, and MATLAB.
  - ASCII: A plain text format used to represent raster data in a grid structure, widely used in ArcGIS, QGIS, and R.

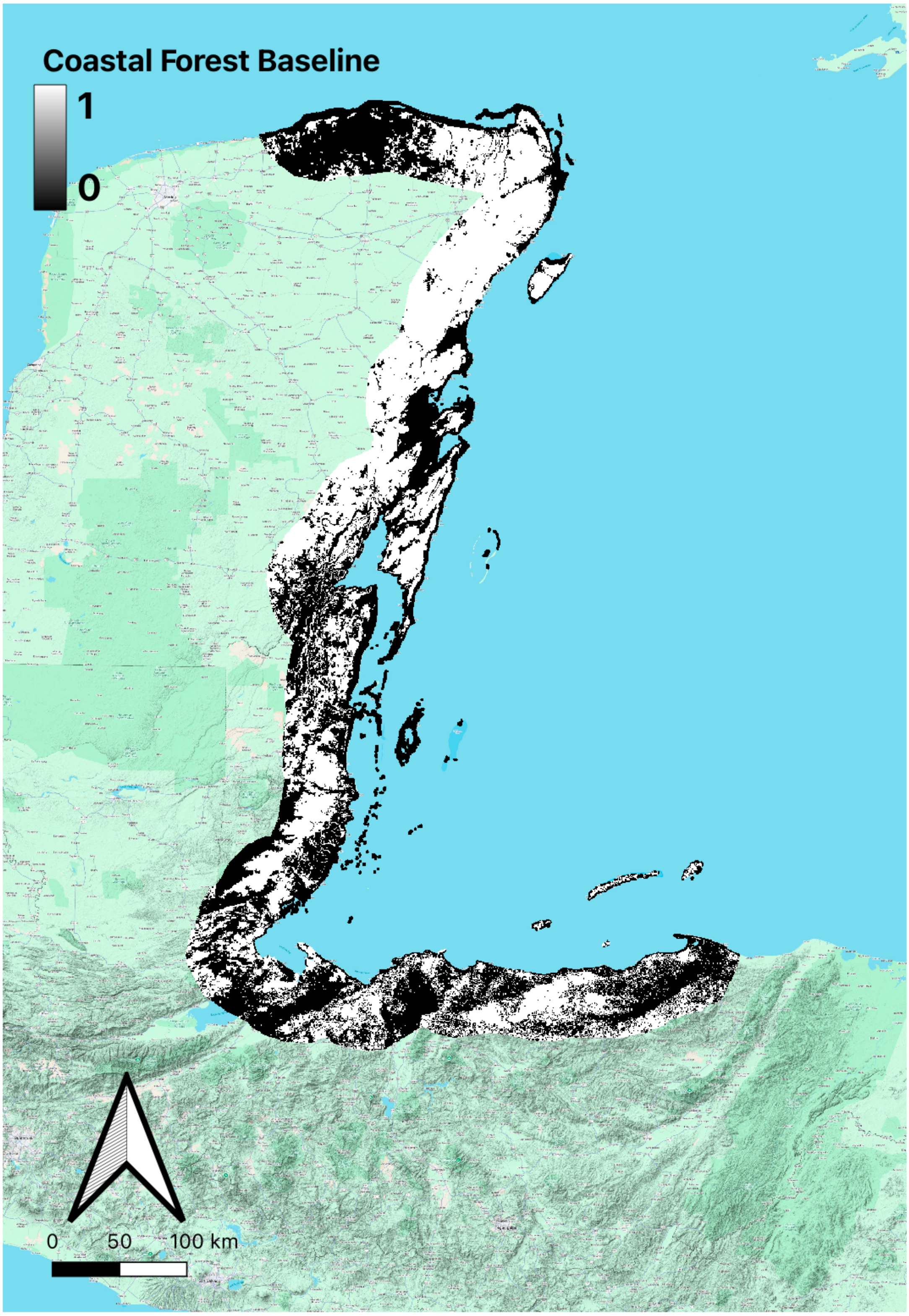


Figure 3. Example of a raster as a GeoTIFF (.tif) representing baseline of coastal forests spanning Belize, Guatemala, Honduras, and Mexico.

2. General Spatial Properties

Spatial datasets must include key spatial properties such as the coordinate reference system (CRS), specifying the datum (e.g., WGS84) and projection (if applicable, e.g., UTM), as well as spatial extent and resolution, to ensure accuracy and usability. The datum defines the Earth's shape and provides a reference for measuring locations on its curved surface. The CRS incorporates the datum and determines how coordinates are expressed, while a projection translates the three-dimensional Earth onto a two-dimensional plane, introducing some distortions. Spatial extent specifies the geographic area covered by the dataset, and for raster data, resolution determines the level of detail based on pixel size. Clearly defining these spatial properties ensures consistency, proper alignment, and meaningful spatial analysis.

- **Defined Coordinate System**: Each spatial dataset must include a clearly defined coordinate system, which includes:
    - **Datum** (e.g., WGS84, NAD83): Defines the shape of the Earth and how locations are measured on its curved surface.
    - **Projection** (e.g., UTM): A mathematical transformation that flattens the Earth's curved surface onto a 2D plane, introducing distortions in area, distance, shape, or direction.
    - Note: not all spatial data need a projection, and in some cases, it is best to work with unprojected data, especially when working with global data or focused on latitude and longitude as opposed to focused on spatial analyses (distance, area, etc.).
- **Defined spatial extent:** Each spatial dataset must include a clearly defined spatial extent which is represented by the minimum and maximum X (longitude) and Y (latitude) coordinates that enclose all the features in a shapefile or data in a raster. The coordinates of the spatial extent are defined in the same units of the CRS and projection.
- **Defined spatial resolution (for rasters specifically):** Each raster file must include a clearly defined resolution, which represents the size of each grid cell or pixel (e.g. 30 m resolution means each pixel represents a 30m x 30m area). The resolution is defined in the same units of the CRS or projection.

3. Attribute Table Standards

An attribute table (Table 1) is a tabular representation of the data associated with spatial features, where rows represent individual features (e.g., points, lines, polygons), and columns contain descriptive attributes about the corresponding spatial features. A well-structured attribute table is crucial for data clarity, enabling users to efficiently analyse and interpret spatial data.

Table 1. Example of an attribute table displaying a subset of coastal communities in Belize.

	SETTNAME	POPULATION	POPSIZE	DISTRICT	TYPE
1	Belize City	5	45584	Belize	City
2	Dangriga	4	8424	Stann Creek	Town
3	Corozal	4	7589	Corozal	Town
4	Punta Gorda	3	4266	Toledo	Town
5	Mango/Independence	3	2881	Stann Creek	Village
6	Ladyville	3	3472	Belize	Village
7	Sarteneja	3	1591	Corozal	Village
8	Ranchito	3	1045	Corozal	Village
9	Seine Bight	2	831	Stann Creek	Village
10	Riversdale	2	685	Stann Creek	Village

- **Appropriate Column Headers**: Use clear, concise, and descriptive column headers that accurately represent the data. Avoid abbreviations unless necessary, and ensure headers are easily understandable by all users.
- **No Spaces in Headers**: Replace spaces in column headers with underscores (_) for consistency.
  - The use of underscores (_) or periods (.) instead of spaces is important for maintaining consistency and compatibility across various software, programming languages, and data processing tools (Figure 4). Many systems, especially those used in data analysis (e.g., R), can misinterpret spaces in column headers, possibly leading to errors or additional pre-processing requirements.
- **Consistent Letter Case**: Use a consistent case format for column headers (e.g., all lowercase).
  - Using a consistent case format (e.g., all lowercase) for column headers enhances readability and reduces the likelihood of errors during data processing (Figure 4). Inconsistent casing can cause confusion and complications when referencing column names in case-sensitive environments (e.g., R).

POOR PRACTICE

lat	LONG	Chlorophyll a	SalinityLevel

BEST PRACTICE

lat	long	chlorophyll_a	salinity_level

Figure 4. Examples of poor (in red) and best (in green) practices for column headers, focusing on underscores and consistent letter cases. Poor practice includes inconsistent capitalization and spacing, while best practice ensures all headers use lowercase letters with underscores replacing spaces

- **Data Column Consistency:** Ensure uniform formatting within each data column (**Error! Reference source not found.**).
    - **Dates:** Use a consistent format, preferably ISO 8601 (YYYY-MM-DD), to avoid ambiguity.
    - **Numeric Values:** Maintain a uniform number of decimal places where applicable.
    - **Text Entries:** Standardize capitalization, spacing, spelling, and abbreviations for categorical data.

POOR PRACTICE

species	date_recorded
bluetang	01-22-2023
Queen Triggerfish	Sept-15-1975
BarJack	04/02/2003

BEST PRACTICE

species	date_recorded
Blue Tang	2023-01-22
Queen Triggerfish	1975-09-15
Bar Jack	2003-04-02

Figure 5. Examples of poor (in red) and best (in green) practices for data columns, emphasizing consistent formatting within columns. Poor practice includes inconsistent formatting (e.g., dates presented in different formats), while best practice ensures uniform formatting across all rows (e.g., dates consistently follow the YYYY-MM-DD format).

- **Complete Data Rows**: Ensure all rows in the attribute table are complete, with no blank cells unless explicitly justified and documented in the metadata (Figure 6). Missing values or data can lead to errors in analysis, misinterpretation of results, or issues when integrating datasets across different platforms and applications.
- **Handling of Missing Data**: Use consistent symbols or values (e.g., "NA", "ND", ”Unknown”,-9999) to represent missing data (Figure 6).
    - **Numeric Fields (for GIS data formats)**: Leave cells blank (null or use a predefined numeric placeholder (e.g., -9999), since text is not supported.
    - **Text or String Fields (for GIS data formats) or Fields in Non-GIS Data Formats:** Use “NA”, “ND” (No Data), or “Unknown” based on context.
        - **"NA" (Not Available)** and **"ND" (No Data")**: Used when data was neither collected, measured, nor recorded. *"NA"* is commonly used in programming languages (e.g., R), while *"ND"* is typically found in spatial datasets.
        - **"Unknown"**: Indicates that a value exists but is not known or identified, suggesting uncertainty rather than absence.

POOR PRACTICE

species	date_recorded	temperature
	2023-01-22	25
Queen Triggerfish	1975-09-15	
Bar Jack	2003-04-02	

BEST PRACTICE

species	date_recorded	temperature
Unknown	2023-01-22	25
Queen Triggerfish	1975-09-15	NA
Bar Jack	2003-04-02	NA

Figure 6. Examples of poor (in red) and best (in green) practices for complete data rows and handling missing data. Poor practice leaves unexplained blanks, while best practice fills missing values with appropriate placeholders (e.g., NA, Unknown) and documents them in the metadata.

Metadata

Metadata provides essential information about a dataset's content, structure, and management, ensuring that data remains discoverable, well-organized, and reusable over time. It enhances transparency, facilitates data interpretation, and supports long-term accessibility. The following guidelines outline best practices for metadata creation and organization, divided into two key areas:

Main Metadata – Encompasses **administrative, descriptive, structural, and preservation details** to provide a comprehensive overview of the dataset (Table 2).

Table 2. Main Metadata, Fields in blue must be included to ensure data clarity, accuracy, and proper integration into GIS systems, while fields in black provide additional context that can improve data usability but are not strictly required.

Field Name	Description
Title	Title of the dataset
Identifier	A system-generated or manually assigned unique code for the dataset
Abstract	Brief description of the dataset
Purpose	Purpose for which the dataset was created
Keywords	Keywords describing the dataset, separated by commas
Topic_Category	ISO Topic Category (e.g., Dredging, Aquaculture, Fishing)
Data_Owner	The organization or individual responsible for the collection, accuracy, and access policies of the dataset.
Data_Owner_Contact_Name	The primary point of contact for the data owner, responsible for managing inquiries, updates, and access requests related to the dataset.
Data_Owner_Contact_Email	The official email address of the data owner’s designated contact person for correspondence regarding data access, corrections, or clarifications.
Data_Owner_Contact_Phone	The phone number of the designated contact person for direct communication regarding the dataset.
Data_Manager	The entity or individual responsible for organizing, curating, storing, and ensuring the quality and integrity of the dataset over time.
Data_Manger_Contact_Name	The primary contact responsible for managing the dataset, including data curation, updates, and integration with other data systems.
Data_Manager_Contact_Email	The email address of the data manager for inquiries regarding data structure, maintenance, and quality assurance.
Data_Manager_Contact_Phone	The phone number of the data manager for direct communication on technical or operational matters related to

Field Name	Description
	data management.
Rights	Intellectual property rights and usage constraints
Access_Constraints	Restrictions on access to the dataset (Public, Restricted, MOU, etc.)
Use_Constraints	Conditions for data usage
Citation	Recommended citation for the dataset in APA format
Data_Format	File format (e.g., Shapefile, GeoTIFF, CSV)
Data_Type	Type of data (Vector, Raster, Tabular)
Spatial_Resolution	Spatial resolution (e.g., 30m, 1km)
CRS	Coordinate Reference System (Datum and Projection; e.g., NAD27 UTM 16N)
Version	Version of the dataset
Version_Control	Version control and updates
Provenance	History and source of data.
Retention_Policy	Policy on data preservation and storage duration.
Preservation_Description	Details on how the dataset is maintained and archived.
Spatial_Extent_Longitude_Min	Min longitude of bounding box
Spatial_Extent_Longitude_Max	Max longitude of bounding box
Spatial_Extent_Latitude_Min	Min latitude of bounding box
Spatial_Extent_Latitude_Max	Max latitude of bounding box
Temporal_Extent_Start	Start date of temporal coverage (YYYY-MM-DD)
Temporal_Extent_End	End date of temporal coverage (YYYY-MM-DD)
Update_Frequency	Frequency of updates (e.g., annual, monthly, as needed)
Date_Created	Date the dataset was created (YYYY-MM-DD)
Date_Modified	Date of last modification (YYYY-MM-DD)
Data_Quality_Statement	Summary of data accuracy, completeness, and limitations.
Processing_Steps	Key transformations or modifications applied to the dataset.
Data_Relationships	Relationships with other datasets or data sources.
File_Dependencies	Dependencies on other datasets or reference files.
Hierarchical_Organization	Organization of files and related datasets.

Attribute Metadata – Focuses on the **technical aspects** of individual data fields, including definitions, data types, and value constraints (Table 3).

Table 3. Attribute Metadata. Fields in blue must be included to ensure data clarity, accuracy, and proper integration into GIS systems, while fields in black provide additional context that can improve data usability but are not strictly required.

Column Name	Description
Attribute_Name	The exact field name as it appears in the shapefile.
Alias	A user-friendly name for the field (alternative to raw field name).
Attribute_Definition	A detailed description of what the field represents.
Attribute_Type	The data type (e.g., Integer, Float, String, Date).
Units_of_Measure	Measurement units (if applicable, e.g., meters, hectares).
Value_Domain	Allowed values or value range (e.g., 0–100, categorical values).
Missing_Value_Handling	Explanation of how missing values are recorded (e.g., NULL, -9999).
Default_Value	The default value assigned if no data is provided.
Constraints	Any restrictions or rules applied to the field (e.g., unique, required).
Attribute_Source	The origin of the data for this field (e.g., field survey, model output).
Attribute_Processing	Any transformations or calculations applied to this field.

Creating or Editing Metadata of a Shapefile

1. Prepare the Metadata in Excel

An Excel sheet can serve as a draft metadata document that can later be used to directly update the metadata of a shapefile. By first organizing metadata in Excel, users can ensure accuracy and completeness before importing it into ArcGIS or QGIS. The metadata excel file contains two sheets: 'main_metadata' and 'attribute_metadata', which define the overall information about the datafile and the attribute-specific details, respectively.

- Carefully complete all fields of the main metadata to provide a comprehensive description of the dataset. Replace the placeholder descriptions in the 'Description' column with precise details specific to the datafile.

- Carefully complete the attribute metadata to provide a description of each attribute (column in the attribute table). Ensure all attribute details, including name, definition, data type, units, and handling of missing data, are correctly filled in.

2. Import or Link the Metadata

All metadata should follow the ISO 19115/19139 standard to ensure alignment with international best practices.

Option 1: Manual Entry via Metadata Editor

To manually enter main metadata for a shapefile in ArcGIS Pro, ArcCatalog, or QGIS, follow these steps:

I. Set metadata style to “ISO 19139 Metadata Implementation”:
   - In ArcGIS Pro, go to Project>Options>Metadata and set metadata style to “ISO 19139 Metadata Implementation Specification” before viewing or editing metadata.
   - In QGIS, the default metadata framework aligns with ISO 19115.

II. Open Metadata:
   - ArcGIS Pro / ArcCatalog:
     - In the Catalog Pane, right-click the shapefile.
     - Select View Metadata.
   - QGIS:
     - In the Layers Panel, right-click the shapefile.
     - Select Properties, then navigate to the Metadata tab.

III. Edit Metadata:
   - In ArcGIS Pro/ArcCatalog, click the Edit Metadata button
   - In QGIS, click the pencil icon (Toggle Editing).
   - Copy and paste relevant fields from the main_metadata excel sheet into the corresponding metadata fields:
     - Title, Abstract, Keywords, Contact Information, Data Format, CRS, Spatial Extent, Temporal Extent, etc.

IV. Save Changes:
   - In ArcGIS Pro / ArcCatalog, click Save.
   - In QGIS, click Apply or OK.

To manually enter attribute metadata (field descriptions, data types, units, and value constraints) for a shapefile in ArcGIS Pro, ArcCatalog, or QGIS, follow these steps:

I. Open Attribute Metadata
   - ArcGIS Pro:
     - In the Catalog Pane, right-click the shapefile.
     - Select Design > Fields to open the Fields View.
   - ArcCatalog:
     - Right-click the shapefile.

        - Select Properties, then navigate to the Fields tab.
    - QGIS:
        - Right-click the layer in the Layers Panel.
        - Select Properties, then navigate to the Fields tab.

II. Edit Attribute Metadata
    - ArcGIS Pro / ArcCatalog:
        - Alias: Enter a clear, human-readable name for each field.
        - Data Type: Confirm or adjust data types (Integer, Float, String, etc.).
        - Domains: Define value constraints for categorical or numerical fields.
        - Allow NULL Values: Specify whether missing values are allowed.
    - QGIS:
        - Toggle Editing Mode (Click the pencil icon).
        - Alias: Enter a Field Alias (a descriptive field name).
        - Constraints: Define value limits, unique values, or required fields.
        - Default Values: Set a default value for missing data.

III. Save Changes
    - ArcGIS Pro / ArcCatalog: Click Save.
    - QGIS: Click Apply or OK.

Option 2: Import Metadata via ArcPy

For efficient bulk metadata import, Python (ArcPy) can automate the process, ensuring accuracy and consistency across multiple fields within the metadata. Refer to Annex 1 - Arcpy script to import main metadata into ArcGIS. and Annex 2 - Arcpy script to import attribute metadata into ArcGIS. for Python script to import main and attribute metadata, respectively.

3. Export Metadata to XML

Once metadata has been manually entered or imported into ArcGIS Pro, ArcCatalog, or QGIS, it should be exported as an ISO 19139-compliant XML file to ensure standardization and interoperability.

To export metadata to XML in ArcGIS Pro and ArcCatalog follow these steps:

I. Open the Metadata Export Tool
    - ArcGIS Pro:
        - Open Catalog Pane and right-click the shapefile or feature class.
        - Select Export Metadata.
    - ArcCatalog:
        - Right-click the dataset.
        - Select Export Metadata.

II. Choose Export Format
    - ArcGIS Pro:
        - Open Catalog Pane and right-click the shapefile or feature class.
        - Select Export Metadata.
    - ArcCatalog:
        - Right-click the dataset.

            - Select Export Metadata.

III. Validate the Exported XML (Optional)
   - Open the exported .xml file in a text editor (Notepad++, VS Code) to check formatting.
   - Use GeoNetwork or ArcGIS Metadata Toolkit for validation.

To export metadata to XML in QGIS follow these steps:

I. Open Layer Properties
   - Right-click the layer in the Layers Panel.
   - Select Properties.
   - Go to the Metadata tab.

II. Export Metadata
   - Click Export Metadata.
   - Choose ISO 19139 format.
   - Select a destination and filename.
   - Click Save.

⚠️ Shapefiles have limited metadata support, meaning that only certain fields are retained within ArcGIS metadata for both attribute metadata (field-specific details) and main metadata (dataset-level information). To ensure comprehensive metadata documentation, it is recommended to store full metadata externally in complementary files, such as a .txt or .csv file, alongside the shapefile.

Quality Assurance and Error Checking

Ensuring the accuracy, consistency, and reliability of spatial data is critical to prevent misinterpretation, support decision-making, and maintain data integrity. Proper quality assurance (QA) and error-checking processes help identify and correct issues before data is published, ensuring it meets defined accuracy and usability standards.

1. Consistency between Datafile and Metadata

Maintaining consistency between the dataset and its metadata is essential for data integrity, usability, and interoperability. Validation should confirm that all elements in the data file align with metadata specifications, reducing errors and ensuring seamless integration into GIS and data management systems.

- **Column Headers:** Ensure field names in the dataset match those documented in metadata.
- **Data Types:** Confirm each attribute's data type (Integer, Float, String, Date) corresponds to metadata definitions.
- **Value Ranges:** Verify that numeric fields fall within defined ranges (e.g., 0–100 for percentage values).
- **Handling of Blanks:** Ensure missing values are handled as documented in metadata (e.g., NULL, -9999).
- **Spatial Extent:** Confirm that the dataset's geographic boundaries align with the documented spatial extent.

2. Attribute Validation

Attribute validation ensures that tabular data (data in the attribute table) aligns with data format and quality standards (see Data Format and Quality). Effective attribute validation requires direct engagement with the dataset, its associated metadata, and collaboration with the data collector or creator to ensure accuracy and consistency. Key attribute validation checks include:

- **Check for Formatting Errors:** Ensure that text, numeric, and date fields follow **standardized formats** (e.g., dates in YYYY-MM-DD, numeric values within appropriate ranges).
- **Enforce Categorical Data Consistency:** Validate that categorical fields use predefined and consistent classification values (e.g., "Protected", "Unprotected", rather than "protected", "UnProtected", "Not Protected").
- **Detect Null or Incorrect Values:** Identify and document missing, unknown, or incorrect attribute values, ensuring all required fields are populated.
- **Apply Logical Consistency Checks:** Ensure values make sense in context (e.g., an elevation field should not contain negative values unless applicable).

3. Topology Error Checking

For vector data, topology rules establish and enforce spatial relationships between features, ensuring geometric consistency, logical accuracy, and seamless data integration. Detecting and correcting topology errors is essential to prevent misalignment, spatial inconsistencies, and analytical inaccuracies that could impact mapping, modelling, and decision-making. Common topology errors and their solutions include:

- **Overlapping Polygons:** Ensure that adjacent polygons do not overlap unless explicitly required (e.g., marine jurisdictional boundaries).
- **Gaps Between Polygons:** Check for and correct unwanted gaps (slivers) between adjacent polygons.
- **Self-Intersecting Features:** Identify and resolve self-intersecting lines or polygons that cause spatial inconsistencies.
- **Unclosed Polygons:** Ensure that polygons are fully closed and valid, preventing errors in area calculations.
- **Duplicate Vertices and Overshoots:** Clean up unnecessary vertices and snap misaligned features to correct positions.

Checking and Correcting Topological Errors
ArcGIS Pro and QGIS provide topology validation tools that detect and correct errors such as overlapping polygons, gaps, duplicate features, and self-intersections.
To detect and correct topology errors within a polygon shapefile in ArcGIS Pro, follow these steps:

I. Create a Topology Rule (Feature Class in a Geodatabase). Note: Topology rules can only be created for feature classes within a geodatabase. Shapefiles do not support topology rules.
    - Open ArcGIS Pro.
    - In the Catalog Pane, right-click the geodatabase.
    - Select New > Topology.
    - In the Create Topology Wizard:
        - Select the feature dataset.
        - Assign a Cluster Tolerance (default is recommended).
        - Click Next.

II. Add Topology Rules
    - In the Topology Properties window, click Add Rule.
    - Select a Feature Class and a Rule Type:
        - Must Not Overlap (for polygons).
        - Must Not Have Gaps (for adjacent polygons).
        - Must Not Self-Intersect (for polylines).
        - Must Not Overlap with (ensures separate feature classes do not overlap).
        - Click OK, then Finish.

III. Validate and Identify Errors
    - Open Geoprocessing Toolbox.
    - Search for Validate Topology and run the tool.
    - Open the Topology Layer and inspect Error Markers.

- Right-click an error and select Fix Error.

To detect and correct topology errors within a polygon shapefile in QGIS, follow these steps:

I. Enable Snapping to Prevent Errors in QGIS
   - Enable Editing Mode:
     - Right-click the layer in the Layers Panel.
     - Select Toggle Editing.
   - Open the Snapping Toolbar:
     - Click View > Toolbars > Snapping Toolbar (if not visible).
   - Configure Snapping Options:
     - Click the Snapping Options (Magnet Icon).
     - Choose Snapping Mode:
       - Vertex Snapping (snap to feature vertices).
       - Edge Snapping (snap to existing feature edges).
       - Intersection Snapping (snap to intersections).
   - Set Snapping Tolerance:
     - Define tolerance in pixels or map units.
     - A smaller tolerance ensures precision, while a larger tolerance ensures broader snapping.

⚠️ Important: Snapping helps when manually adjusting errors detected by Topology Checker.

II. Check for Topological Errors using the Topology Checker Tool
   - Enable the Topology Checker Plugin:
     - Go to Plugins > Manage and Install Plugins.
     - Search for Topology Checker and install it.
   - Run Topology Checker:
     - Click Processing Toolbox > Topology Checker.
     - Click Configure Rules and select Add Rule.
     - Choose a rule type such as:
       - Must Not Overlap (for polygons).
       - Must Not Have Gaps (for adjacent polygons).
   - Click Run to identify errors.

III. Fix Topological Errors
   - Overlapping Polygons> Merge Tool/Reshape Tool
   - Gaps between Polygons> Fill Gaps Tool/Vertex Tool
   - Self-Intersecting Lines>Split Line Tool
   - Misaligned Vertices> Split Line Tool

4. Duplicates

Duplicate features in spatial datasets can lead to data redundancy, analytical errors, and misrepresentation of spatial patterns. Identifying and removing duplicates ensures data integrity, improves processing efficiency, and enhances the accuracy of spatial analyses.

- **Identify Duplicate Records in Attribute Tables:** In tabular data (or the attribute table) use unique identifiers to detect duplicate rows and compare spatial and non-spatial attributes to find duplicated entries.
- **Detect Overlapping or Redundant Geometries:** Check if multiple polygons, lines, or points occupy the same spatial location when only one should exist. Use GIS duplicate detection tools (e.g., QGIS's "Check for Duplicates" plugin, ArcGIS's "Find Identical" tool).
- **Standardize Naming Conventions:** Ensure identical features do not appear under multiple inconsistent names (e.g., "MPA Belize North" vs. "MPA_North_Belize").

5. Detecting and Removing Duplicates

To detect and remove duplicates in ArcGIS, follow these steps:

I. Identify Duplicate Records in Attribute Tables
   - Open the Attribute Table.
   - Identify duplicate records using unique identifiers:
     - Right-click the Field Header (e.g., ID, Name).
     - Sort the column Ascending or Descending.
     - Look for repeated entries manually.
     - Manually delete duplicate records.
   - Alternatively, use Select by Attributes:
     - Open Select by Attributes.
     - Use SQL expressions to count the number of rows corresponding to each unique 'Field Name' below.
     - Click Run to highlight duplicates.
     - Click Delete Selected Records.

```
"Field_Name" IN (SELECT Field_Name FROM Table GROUP BY Field_Name HAVING COUNT(*) > 1)
```

For easy copying and pasting into an SQL query:
"Field_Name" IN (SELECT Field_Name FROM Table GROUP BY Field_Name HAVING COUNT(*) >1)

II. Detect Overlapping or Redundant Geometries
   - Using the Geoprocessing Toolbox.
     - Open Geoprocessing Toolbox.
     - Search for Find Identical.
     - Set parameters:
       - Input Features: Select your dataset.
       - Fields to Compare: Choose 'Shape' for spatial duplicates and 'ID' for attribute duplicates.
     - Click Run.
     - Results:
       - Duplicates are flagged in a new field (Join_Count).
       - Export the results to a new layer to analyze them.
   - Using the "Delete Identical" Tool

- Open Geoprocessing Toolbox.
- Search for Delete Identical.
- Select the dataset and define duplicate criteria:
  - Shape to remove geometrically identical features.
  - Attribute fields to remove identical non-spatial duplicates.
- Click Run.

To detect and remove duplicates in QGIS, follow these steps:

I. Identify Duplicate Records in Attribute Tables
   - Using the "Remove Duplicate by Attribute" Tool.
     - Open the Processing Toolbox.
     - Search for Remove Duplicate by Attribute.
     - Select:
       - Input Layer: Choose the dataset.
       - Attribute to Compare: Choose fields that define uniqueness.
     - Click Run.
     - Results:
       - The tool creates a new layer with duplicates removed.
   - Using "Select by Expression" and Manual Deletion
     - Open Attribute Table.
     - Click Select by Expression.
     - Use this expression to detect duplicates
     - Click Select > Delete Selected Features.

```
count("Field_Name") OVER (PARTITION BY "Field_Name") > 1
```

For easy copying and pasting into an SQL query:
count(“Field_Name”) OVER (PARTITION BY “Field_Name”)>1

II. Detect Overlapping or Redundant Geometries
   - Using "Delete Duplicate Geometries" Tool
     - Open Processing Toolbox.
     - Search for Delete Duplicate Geometries.
     - Select:
       - Input Layer: Choose the dataset.
       - Tolerance: Set a small threshold (e.g., 0.0001).
     - Click Run to delete overlapping features.

6. Verifying Spatial Relationships

Spatial relationships determine how geographic features interact and relate to one another within a dataset. Proper validation ensures that these relationships are logically accurate, consistent, and free from spatial errors, preventing misinterpretation and ensuring that analyses, decision-making, and spatial modelling are based on reliable data.

- **Check for Correct Feature Adjacency:** Ensure that polygons that should share boundaries do so without gaps or overlaps.
- **Confirm Layer Alignment and Logical Consistency:** Ensure that all spatial layers **align correctly** without unexpected shifts (e.g., marine resource use layers should match administrative boundaries). Check for logical consistencies (e.g. water bodies don't overlap with land)

7. Data Completeness Assessment

A dataset is considered complete when it includes all required information, with no missing records or gaps that could compromise its accuracy and usability. Any intentional omissions or missing data should be clearly documented, ensuring transparency and enabling users to understand the dataset's limitations and constraints.

- **Identify missing fields:** Ensure that all required attributes are present and correctly populated.
- **Check for Gaps in Spatial Coverage:** Verify that no areas or regions are missing from the dataset where data is expected.
- Document justifiable omissions: Some missing data may be valid (e.g., incomplete survey data due to access constraints). Ensure such gaps are documented in metadata.

Detecting and Correcting Features without Spatial Information

To detect features without spatial information in ArcGIS Pro, follow these steps:

I. Open the Attribute Table.
II. Click Select by Attributes.
III. Use this SQL query to select features missing geometry:

```
Shape IS NULL
```

For easy copying and pasting into an SQL query:
Shape is NULL

IV. Click Run.
If records are selected, they lack spatial data.

To detect features without spatial information in QGIS, follow these steps:

I. Open **QGIS** and load the dataset.
II. Open the **Attribute Table**.
III. Click **Select by Expression** (Ctrl + F3).

IV. Use this expression to identify features without geometry:

```
$geometry IS NULL
```

For easy copying and pasting into an SQL query:
$geometry IS NULL

V. Click **Run**.

If records are selected, they lack spatial data.

Whether you are working in ArcGIS or QGIS, ensure they are not duplicates of other features with valid geometry (refer to the Duplicates section). If the missing spatial data is critical and in fact not duplicates, collaborate with the data provider to obtain the necessary coordinates. If retrieval is not possible, consider omitting the records while clearly documenting the omission in the metadata for transparency.

Data Curation Recommendations

Up to this point, the guidelines have outlined technical data curation processes to ensure accuracy, completeness, and integrity. Moving forward, the focus shifts from technical protocols to best practices for data storage, organization, and long-term curation. These recommendations are designed to enhance data accessibility and usability, ensuring its value for future users.

Data Archiving and Storage

Effective data archiving is essential for preserving datasets, ensuring long-term accessibility, and maintaining data integrity. A well-organized and secure archiving system supports informed decision-making, facilitates collaboration, and preserves valuable data for future research, policy development, and operational needs. The following guidelines provide a framework for implementing robust data preservation practices to maintain data integrity and longevity.

1. Systematic Data Organization

Organizing data systematically is critical for ensuring it is easy to locate, access, and use. Proper file labelling, version control, and timestamps ensure that data updates are tracked, reducing errors and maintaining data integrity over time.

- **File Labelling and Naming Conventions:** Use consistent, structured naming conventions to enhance searchability and avoid confusion. Avoid spaces in file and folder names; use underscores (_) or hyphens (-) instead. Ensure file names are descriptive but not excessively long.
    - Include key information in filenames if applicable:
        - **Project Name or Acronym** (e.g., ICZMP)
        - **Subcategory** (e.g. aquaculture_farms, fishing_areas)
        - **Date in YYYY-MM-DD format for sorting** (e.g., 2024_02_10)
        - **Version number** (e.g., _v1, _v2, _v3)
        - **Data type** (e.g., _metadata, _survey)

Example:

The shapefile representing fishing areas in Belize (Figure 1) could be named:

ICZMP_fishing_areas_2016.shp

- **Version Control:** Implement a clear version control strategy to track changes and avoid duplication. Use numerical or descriptive versioning and potentially store older versions in an Archive folder to prevent clutter.
    - Numerical Versioning: v1, v2, v3 (increment as changes are made)
    - Descriptive Versioning: draft, final
    - Avoid ambiguous terms like "new" or "latest."

- **Hierarchical Data Organization:** Establish a clear and coherent hierarchical folder structure that reflects logical groupings with metadata corresponding to each data file for easy navigation and accessibility. Avoid deeply nested folders that can make it difficult to find and manage files.
    - **Core Folder Structure**: A standardized folder structure should be predictable and intuitive. Group files based on project names, datasets, years, or relevant categories to streamline retrieval. Common top-level categories for a given project or dataset include: raw data, processed data, metadata, and archive.
- **Structural Metadata:** Each folder should include structural metadata to describe its contents, purpose, and organization. Structural metadata provides context for data management, ensuring files are findable, accessible, interoperable, and reusable (FAIR). Every folder should maintain a structured metadata file (.xml, .txt, json) that captures folder structure and organization, file format specifications, versioning details, and relationships between files (see Metadata).

2. Robust Data Storage Strategies

Effective data preservation ensures that valuable datasets remain accessible, secure, and usable over the long term. By implementing strategies such as standardized file formats, appropriate data storage systems, and regular backups can protect data from loss, corruption, and obsolescence.

- **Standardized File Formats**: Store data in widely accepted, non-proprietary formats (e.g., CSV, GeoTIFF, Shapefiles) to ensure compatibility and long-term usability.
- **Define Data Storage System:** Define storage requirements based on data sensitivity levels (e.g., public, restricted, confidential). Design storage systems to be secure and capable of accommodating data growth efficiently while maintaining performance and integrity.
    - **Public Data** can be stored in a data repository or cloud storage, depending on how the data is managed and accessed:
        - **Data Repository (e.g., CKAN, GeoNode)**: Best for structured, long-term data management where metadata, version control, and advanced searchability are needed. Ideal for datasets that will be frequently accessed, analysed, or updated. A website can serve as the user-friendly front-end, linking to the repository for external users to easily search, preview, and download data.
        - **Cloud Storage (e.g., Google Drive, OneDrive)**: Suitable for simple file hosting and sharing, where structured metadata and version control are not a priority. Easier for quick access to downloadable files but lacks structured dataset management.
    - **Restricted or confidential data** can be stored in a secure data repository, local secure servers onsite, and/or private encrypted cloud storage, depending on access needs and security requirements.
        - **Secure Data Repository**: Best for structured datasets requiring metadata management, version control, and controlled access.
        - **Local Secure Servers**: Ideal for highly sensitive data that must remain on-premises for maximum security and legal compliance.

        - **Private Encrypted Cloud Storage**: Suitable for scalable storage, offsite backups, and controlled external access for authorised partners with strict encryption and authentication protocols.
- **Data Retention Policies**: Define and follow clear policies on how long data will be retained, archived, or deleted.
- **Regular Backups**: Schedule regularly automated (e.g., weekly, monthly, etc.) and secure backups to prevent data loss. Backups should use a mix of storage types, including those already outlined above, such as cloud platforms, secure data repositories, and local servers, as well as commonly used physical storage devices. Examples include:

    - **External Hard Drives and Solid State Drives (SSDs):** Ideal for on-site, physical backups. SSDs are faster and more resilient to physical damage, while traditional hard drives offer high capacity at lower cost.
    - **Network-Attached Storage (NAS):** Suitable for team environments needing shared access and redundancy.
    - **Cloud Storage and Secure Servers:** As previously noted, cloud platforms (e.g., Google Drive, OneDrive) offer convenient off-site backup options, while secure data repositories and encrypted servers ensure backups of sensitive datasets remain protected.

    To maximize data protection, follow the **3-2-1 rule**: keep at least **three copies** of your data, on **two different storage media**, with **one stored off-site**.

Data Security and Sharing

Clear data security and sharing protocols ensure that spatial data is distributed efficiently, securely, and in compliance with organizational policies. The following guidelines outline best practices for managing data requests and access.

1. Data Security and Internal Data Sharing

Implementing robust data security measures is critical for protecting sensitive or restricted information and ensuring compliance with organizational policies and legal standards. Data security encompasses both technical safeguards and administrative processes, such as access controls, secure storage, and formal agreements, to maintain confidentiality, integrity, and availability of data.

- **Clarify Data Classification Levels:** Define different levels of data sensitivity (e.g., public, restricted, confidential) and outline corresponding security measures. Specify any conditions under which restricted data may be reclassified for broader access.
- **Authorization Protocols**: Establish clear guidelines specifying who is authorized to access, view, or modify specific datasets. Implement role-based access control (RBAC) to ensure that users only have access to the data necessary for their role, minimizing the risk of unauthorized access or misuse.
- **Access Review and Audit Mechanisms:** Establish periodic reviews of user access to ensure that permissions remain appropriate. Conduct regular security audits and compliance checks.
- **Encryption for Storing and Transmission:** Use encryption for storing and transmitting sensitive data. While a VPN secures network traffic, files should also be

encrypted when stored or shared externally. Use secure file transfer methods such as end-to-end encryption to protect data beyond the VPN environment.
- **Incident Response and Data Breach Protocols:** Outline steps to take in case of a data breach, unauthorized access, or loss of data integrity. Assign roles for handling security incidents, notifications, and mitigation strategies.
- **Backup Protocols:** Ensure regular backups of critical datasets and outline recovery procedures in case of data loss.

2. Public Data Availability and Maintenance

Publicly available data should be readily accessible, well-documented, and updated regularly to ensure reliability and broad usability.

- **Publishing Public Datasets:** Public datasets should be regularly updated and made accessible through relevant regional and national data-sharing platforms to ensure accuracy, consistency, and widespread availability.
- **Open Data Platforms:** Identify platforms where datasets should be hosted (e.g., CZMAI website, government open data portals, regional GIS hubs, national/regional data sharing platforms).
- **Embargo Policies**: Some datasets may require an embargo period before public release to allow for the data provider sufficient time to review, validate, and finalize publication on their end. Embargo periods should be clearly documented with expected release timelines.
- **Update Frequency:** Establish a schedule for regular updates to maintain data relevance.
- **Metadata Requirements:** Ensure comprehensive metadata is provided covering administrative, descriptive, structural, technical, and preservation metadata. Include all core fields to ensure data clarity, accuracy (see Metadata).

3. Restricted Data Sharing Protocols

Restricted data requires controlled access and formal agreements before sharing. Access should be carefully managed to protect confidentiality and prevent misuse.

- **Terms of Use**: For restricted and confidential data, require data users to agree to detailed terms of use or licensing agreements before accessing data. These agreements should outline acceptable use, restrictions, and obligations to safeguard sensitive information, and ways to reference data.
- **Data Sharing Agreements & Non-Disclosure Agreements**: Require all data users to access restricted datasets to sign Non-Disclosure Agreements (NDAs) or formal data-sharing agreements. These agreements should specify:
    - Data usage limitations.
    - Restrictions on sharing or reproducing the data.
    - Data destruction policies upon project completion or agreement termination.
- **Secure File Transfer Methods:** Use secure file transfer methods (e.g., Virtual Private Network (VPN), Secure File Transfer Protocol (SFTP), or encrypted email) for

transferring restricted or confidential data. Avoid sending sensitive datasets over unencrypted platforms like standard email or cloud drives.

- **Access Logs and Monitoring**: Maintain detailed logs of data access, modifications, and downloads.

4. Data Request and Access Monitoring

Providing a clear and efficient process for data requests ensures an efficient and transparent process for users. Whether through a designated point of contact or a centralized request portal, these methods simplify communication and streamline the submission and fulfilment of data requests.

- **Point of Contact:** Include a designated email contact of the director and the data manager (or a generic email for continuity) in metadata for datasets available upon request**.** To streamline the data sharing request process, instructions should be provided of how data users should first engage with the point of contact
- **Centralized Data Request Portal:** Alternatively, include a link to a dedicated data request portal that streamlines and automates the request process. Design the data request portal to collect essential information from users.
- **Information Required from Requesters:** To streamline the data sharing request process (point of contact or centralized data request portal), key information should be collected from requesters to better understand the nature, use, and technicalities of the data request, including:
    - **Name and Affiliation**: For record-keeping and identifying the requestor.
    - **Specific Dataset Requested**: Clearly identify the data being requested.
    - **Spatial and Temporal Extent**: Details on the geographic area and time period of interest.
    - **Scale and Resolution**: Any specifications regarding scale or resolution of data.
    - **Intended Use**: A brief description of how the data will be used (e.g., research, policy development, education).
    - **Point of Contact for the MOU** (if applicable and distinct from the data requester): Specify the individual responsible for handling MOU-related agreements and data access, including:
        - **Name**
        - **Position**
        - **Email**

5. Data Request and Access Monitoring

An organized and transparent data request and tracking system, combined with clear and timely communication, is essential for efficient data sharing. Systematic tracking ensures accountability and compliance, while prompt responses enhance user experience and project efficiency.

- **Data Request Tracking System:** Implement a system to track all incoming data requests, including the date of submission, status (e.g., pending, approved, completed), and resolution.
- **Timely Responses:** Ensure timely responses to data requests, with clear communication if additional time or information is needed.

User Training and Support

Empowering data users with the knowledge and tools to effectively utilize geospatial data is essential for maximizing its usefulness and impact. Providing accessible training and clear support mechanisms ensures that users can navigate datasets confidently, interpret them accurately, and apply them effectively. The following guidelines outline best practices to enhance user engagement and maximize the value of geospatial data.

1. User Support Resources and Accessibility

Providing robust user support ensures that data users can confidently access, interpret, and utilize datasets effectively. Tailored training resources and responsive help desk services empower users of all skill levels to navigate geospatial data efficiently, enhancing their experience and enabling productive outcomes.

- **Tailored Training Sessions**: Create a user-friendly online repository offering structured guides, tutorials, and videos designed to accommodate various skill levels and learning needs. Topics will include data acquisition (particularly when it involves more than simple downloads or requires customization), fundamental data manipulation, interpretation, geospatial analysis, and visualization, ensuring accessible, step-by-step learning.
- **Help Desk Services**: Implement a dedicated support system where users can seek assistance, report issues, or request guidance on data usage. This service may function as the primary point of contact for data sharing or operate as a separate communication channel, ensuring timely and efficient support.

2. Demonstrating Data Value

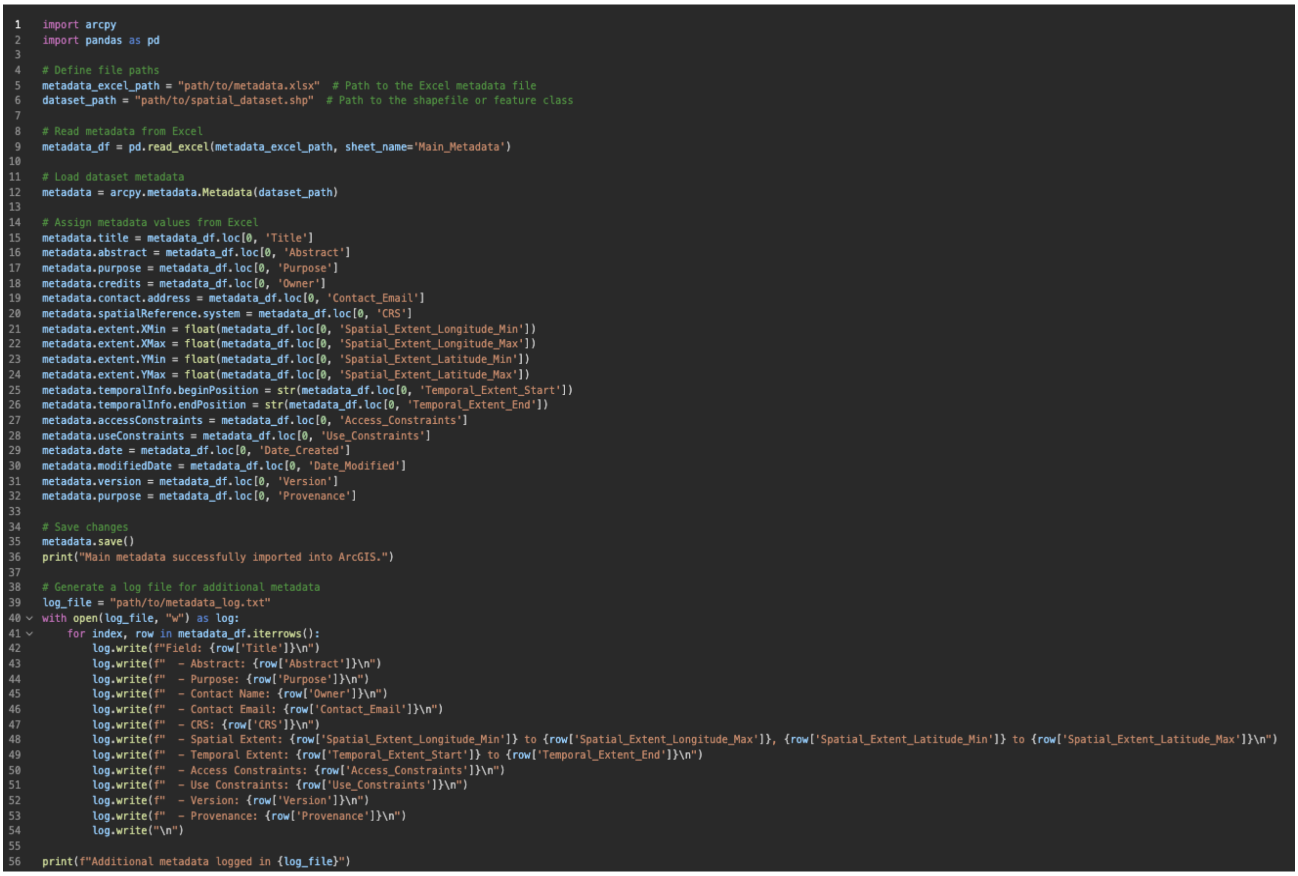
Showcasing the value of data helps users understand its real-world applications and potential, demonstrating its relevance and impact across diverse fields. By providing accessible examples and success stories, users can gain deeper insights into the dataset's significance, empowering them to apply it effectively and uncover new opportunities in their own work.

- **Curated Use Cases**: Compile and publish a collection of reports, projects, and success stories in an easily accessible online repository, showcasing how the data has been effectively applied in various contexts, including policy-making, conservation, and urban planning.

Annexes

Annex 1 - Arcpy script to import main metadata into ArcGIS.

Annex 1. Arcpy script to import main metadata into ArcGIS. Below is a screenshot of the script with useful colour coding, along with a formatted ArcPy code snippet ready for copy and paste. To use this script, update the file paths to match your specific metadata file and dataset.



```python
import arcpy
import pandas as pd

# Define file paths
metadata_excel_path = "path/to/metadata.xlsx"  # Path to the Excel metadata file
dataset_path = "path/to/spatial_dataset.shp"  # Path to the shapefile or feature class

# Read metadata from Excel
metadata_df = pd.read_excel(metadata_excel_path, sheet_name='Main_Metadata')

# Load dataset metadata
metadata = arcpy.metadata.Metadata(dataset_path)

# Assign metadata values from Excel
metadata.title = metadata_df.loc[0, 'Title']
metadata.abstract = metadata_df.loc[0, 'Abstract']
metadata.purpose = metadata_df.loc[0, 'Purpose']
metadata.credits = metadata_df.loc[0, 'Owner']
metadata.contact.address = metadata_df.loc[0, 'Contact_Email']
metadata.spatialReference.system = metadata_df.loc[0, 'CRS']
metadata.extent.XMin = float(metadata_df.loc[0, 'Spatial_Extent_Longitude_Min'])
metadata.extent.XMax = float(metadata_df.loc[0, 'Spatial_Extent_Longitude_Max'])
metadata.extent.YMin = float(metadata_df.loc[0, 'Spatial_Extent_Latitude_Min'])
metadata.extent.YMax = float(metadata_df.loc[0, 'Spatial_Extent_Latitude_Max'])
metadata.temporalInfo.beginPosition = str(metadata_df.loc[0, 'Temporal_Extent_Start'])
metadata.temporalInfo.endPosition = str(metadata_df.loc[0, 'Temporal_Extent_End'])
metadata.accessConstraints = metadata_df.loc[0, 'Access_Constraints']
metadata.useConstraints = metadata_df.loc[0, 'Use_Constraints']
metadata.date = metadata_df.loc[0, 'Date_Created']
metadata.modifiedDate = metadata_df.loc[0, 'Date_Modified']
metadata.version = metadata_df.loc[0, 'Version']
metadata.purpose = metadata_df.loc[0, 'Provenance']

# Save changes
metadata.save()
print("Main metadata successfully imported into ArcGIS.")

# Generate a log file for additional metadata
log_file = "path/to/metadata_log.txt"
with open(log_file, "w") as log:
    for index, row in metadata_df.iterrows():
        log.write(f"Field: {row['Title']}\n")
        log.write(f"  - Abstract: {row['Abstract']}\n")
        log.write(f"  - Purpose: {row['Purpose']}\n")
        log.write(f"  - Contact Name: {row['Owner']}\n")
        log.write(f"  - Contact Email: {row['Contact_Email']}\n")
        log.write(f"  - CRS: {row['CRS']}\n")
        log.write(f"  - Spatial Extent: {row['Spatial_Extent_Longitude_Min']} to {row['Spatial_Extent_Longitude_Max']}, {row['Spatial_Extent_Latitude_Min']} to {row['Spatial_Extent_Latitude_Max']}\n")
        log.write(f"  - Temporal Extent: {row['Temporal_Extent_Start']} to {row['Temporal_Extent_End']}\n")
        log.write(f"  - Access Constraints: {row['Access_Constraints']}\n")
        log.write(f"  - Use Constraints: {row['Use_Constraints']}\n")
        log.write(f"  - Version: {row['Version']}\n")
        log.write(f"  - Provenance: {row['Provenance']}\n")
        log.write("\n")

print(f"Additional metadata logged in {log_file}")
```

Figure 7. Screenshot of python code to use in ArcPy to import main metadata. To use this script, update the file paths to your metadata file and dataset to reflect your specific setup.

Below, is the ArcPy code for importing the main metadata in ArcGIS. You can easily copy and paste it into ArcPy for use. To use this script, update the file paths to your metadata file and dataset to reflect your specific setup. update the file paths to your metadata and data file to match your situation.

import arcpy

import pandas as pd

 # Define file paths

metadata_excel_path = "path/to/metadata.xlsx" # Path to the Excel metadata file

dataset_path = "path/to/spatial_dataset.shp" # Path to the shapefile or feature class

 # Read metadata from Excel

metadata_df = pd.read_excel(metadata_excel_path, sheet_name='Main_Metadata')

 # Load dataset metadata

metadata = arcpy.metadata.Metadata(dataset_path)

 # Assign metadata values from Excel

metadata.title = metadata_df.loc[0, 'Title']

metadata.abstract = metadata_df.loc[0, 'Abstract']

metadata.purpose = metadata_df.loc[0, 'Purpose']

metadata.credits = metadata_df.loc[0, 'Owner']

metadata.contact.address = metadata_df.loc[0, 'Contact_Email']

metadata.spatialReference.system = metadata_df.loc[0, 'CRS']

metadata.extent.XMin = float(metadata_df.loc[0, 'Spatial_Extent_Longitude_Min'])

metadata.extent.XMax = float(metadata_df.loc[0, 'Spatial_Extent_Longitude_Max'])

metadata.extent.YMin = float(metadata_df.loc[0, 'Spatial_Extent_Latitude_Min'])

metadata.extent.YMax = float(metadata_df.loc[0, 'Spatial_Extent_Latitude_Max'])

metadata.temporalInfo.beginPosition = str(metadata_df.loc[0, 'Temporal_Extent_Start'])

metadata.temporalInfo.endPosition = str(metadata_df.loc[0, 'Temporal_Extent_End'])

metadata.accessConstraints = metadata_df.loc[0, 'Access_Constraints']

metadata.useConstraints = metadata_df.loc[0, 'Use_Constraints']

metadata.date = metadata_df.loc[0, 'Date_Created']

metadata.modifiedDate = metadata_df.loc[0, 'Date_Modified']

metadata.version = metadata_df.loc[0, 'Version']

metadata.purpose = metadata_df.loc[0, 'Provenance']

 # Save changes

metadata.save()

print("Main metadata successfully imported into ArcGIS.")

 # Generate a log file for additional metadata

log_file = "path/to/metadata_log.txt"

```
with open(log_file, "w") as log:

    for index, row in metadata_df.iterrows():

        log.write(f"Field: {row['Title']}\n")

        log.write(f"  - Abstract: {row['Abstract']}\n")

        log.write(f"  - Purpose: {row['Purpose']}\n")

        log.write(f"  - Contact Name: {row['Owner']}\n")

        log.write(f"  - Contact Email: {row['Contact_Email']}\n")

        log.write(f"  - CRS: {row['CRS']}\n")

        log.write(f"  - Spatial Extent: {row['Spatial_Extent_Longitude_Min']} to
{row['Spatial_Extent_Longitude_Max']}, {row['Spatial_Extent_Latitude_Min']} to
{row['Spatial_Extent_Latitude_Max']}\n")

        log.write(f"  - Temporal Extent: {row['Temporal_Extent_Start']} to
{row['Temporal_Extent_End']}\n")

        log.write(f"  - Access Constraints: {row['Access_Constraints']}\n")

        log.write(f"  - Use Constraints: {row['Use_Constraints']}\n")

        log.write(f"  - Version: {row['Version']}\n")

        log.write(f"  - Provenance: {row['Provenance']}\n")

        log.write("\n")

print(f"Additional metadata logged in {log_file}")
```

Annex 2 - Arcpy script to import attribute metadata into ArcGIS.

Annex 2. Arcpy script to import attribute metadata into ArcGIS. Below is a screenshot of the script with useful colour coding, along with a formatted ArcPy code snippet ready for copy and paste. To use this script, update the file paths to match your specific metadata file and dataset.

```python
import arcpy
import pandas as pd

# Define paths
shapefile_path = "path/to/shapefile.shp"
metadata_excel_path = "path/to/metadata.xlsx"

# Read metadata from Excel
metadata_df = pd.read_excel(metadata_excel_path, sheet_name='Attributes')

# Start Editing Session
arcpy.env.workspace = shapefile_path
arcpy.MakeFeatureLayer_management(shapefile_path, "temp_layer")

# Loop through each field in the Excel metadata
for index, row in metadata_df.iterrows():
    field_name = row['Attribute_Name']
    field_type = row['Attribute_Type']

    # ArcGIS field types mapping
    field_type_mapping = {
        "Integer": "LONG",
        "Float": "DOUBLE",
        "String": "TEXT",
        "Date": "DATE"
    }

    # Convert to ArcGIS field type
    arcgis_field_type = field_type_mapping.get(field_type, "TEXT")  # Default to TEXT if unknown

    # Check if the field exists in the shapefile
    existing_fields = [f.name for f in arcpy.ListFields(shapefile_path)]

    if field_name not in existing_fields:
        # Add field if it doesn't exist
        arcpy.AddField_management(shapefile_path, field_name, arcgis_field_type)
        print(f"Added field: {field_name} ({arcgis_field_type})")
    else:
        print(f"Field '{field_name}' already exists. Skipping...")

# Cleanup
arcpy.Delete_management("temp_layer")
print("Attribute metadata import completed.")

# Generate a log file for additional metadata
log_file = "path/to/metadata_log.txt"
with open(log_file, "w") as log:
    for index, row in metadata_df.iterrows():
        log.write(f"Field: {row['Attribute_Name']}\n")
        log.write(f"  - Definition: {row['Attribute_Definition']}\n")
        log.write(f"  - Type: {row['Attribute_Type']}\n")
        log.write(f"  - Units: {row['Units_of_Measure']}\n")
        log.write(f"  - Value Domain: {row['Value_Domain']}\n")
        log.write(f"  - Missing Value Handling: {row['Missing_Value_Handling']}\n")
        log.write("\n")

print(f"Additional metadata logged in {log_file}")
```

Figure 8. Screenshot of python code to use in ArcPy to import attribute metadata. To use this script, update the file paths to your metadata file and dataset to reflect your specific setup.

Below is the ArcPy code for importing the attribute metadata in ArcGIS. You can easily copy and paste it into ArcPy for use. To use this script, update the file paths to your metadata file and dataset to reflect your specific setup. update the file paths to your metadata and data file to match your situation.

```
import arcpy

import pandas as pd


# Define paths

shapefile_path = "path/to/shapefile.shp"

metadata_excel_path = "path/to/metadata.xlsx"


# Read metadata from Excel

metadata_df = pd.read_excel(metadata_excel_path, sheet_name='Attributes')


# Start Editing Session

arcpy.env.workspace = shapefile_path

arcpy.MakeFeatureLayer_management(shapefile_path, "temp_layer")


# Loop through each field in the Excel metadata

for index, row in metadata_df.iterrows():

    field_name = row['Attribute_Name']

    field_type = row['Attribute_Type']


    # ArcGIS field types mapping

    field_type_mapping = {

        "Integer": "LONG",

        "Float": "DOUBLE",

        "String": "TEXT",
```

```
        "Date": "DATE"
    }

    # Convert to ArcGIS field type
    arcgis_field_type = field_type_mapping.get(field_type, "TEXT")  # Default to TEXT if unknown

    # Check if the field exists in the shapefile
    existing_fields = [f.name for f in arcpy.ListFields(shapefile_path)]

    if field_name not in existing_fields:
        # Add field if it doesn't exist
        arcpy.AddField_management(shapefile_path, field_name, arcgis_field_type)
        print(f"Added field: {field_name} ({arcgis_field_type})")
    else:
        print(f"Field '{field_name}' already exists. Skipping...")

# Cleanup
arcpy.Delete_management("temp_layer")
print("Attribute metadata import completed.")

# Generate a log file for additional metadata
log_file = "path/to/metadata_log.txt"
with open(log_file, "w") as log:
    for index, row in metadata_df.iterrows():
        log.write(f"Field: {row['Attribute_Name']}\n")
        log.write(f"  - Definition: {row['Attribute_Definition']}\n")
        log.write(f"  - Type: {row['Attribute_Type']}\n")
```

```
    log.write(f"  - Units: {row['Units_of_Measure']}\n")
    log.write(f"  - Value Domain: {row['Value_Domain']}\n")
    log.write(f"  - Missing Value Handling: {row['Missing_Value_Handling']}\n")
    log.write("\n")

print(f"Additional metadata logged in {log_file}")
```